PROCESSAMENTO PARALELO DE CONSULTAS DE APOIO À DECISÃO EM GRIDS

Nelson Peixoto Kotowski Filho

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof a . Marta Lima de Queirós Mattoso, D.Sc.

Prof. Alexandre de Assis Bento Lima, D.Sc.

Prof. Geraldo Zimbrão da Silva, D.Sc.

Prof a . Carla Osthoff Ferreira de Barros, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

MARÇO DE 2008

KOTOWSKI FILHO, NELSON PEIXOTO

Processamento Paralelo de Consultas de Apoio à Decisão em Grids [Rio de Janeiro] 2008

XI, 93 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2008)

Dissertação – Universidade Federal do Rio de Janeiro, COPPE

1. Grades Computacionais

2. Bancos de Dados Distribuídos

3. Processamento Paralelo de Consultas Distribuídas

I. COPPE/UFRJ II. Título (série)

A todos,

"Eis que é chegado o momento de reunir tudo aquilo que juntaste e dizer: é fim de uma etapa e início de uma nova caminhada."

Agradecimentos

Que estas palavras permaneçam para sempre como retrato de uma época única, valorosa e triunfante.

À minha professora e orientadora Marta Mattoso, que me acolheu, ouviu e chamou minha atenção para este universo de conhecimento e trabalho que a universidade representa. Suas palavras sempre firmes e objetivas ficarão como exemplo de comportamento, sabedoria e paciência.

Alexandre, a você, meu co-orientador, fica também o meu agradecimento e um muito obrigado por ter escutado todas as dúvidas durante este curso de Mestrado. Sua tranqüilidade foi essencial para a conclusão desta dissertação. Os frutos desta dissertação não ocorreriam sem sua participação e incentivo.

Meu sincero agradecimento a todos que compõem a banca. É através de vocês que posso enriquecer ainda mais esta dissertação.

Ao professor Patrick Valduriez e à professora Esther Pacitti, que também me receberam de braços abertos e me acompanharam com excelentes idéias e contribuições durante todo o curso e pela oportunidade de acesso ao *Grid'5000*.

A toda equipe do *Grid'5000*, que me escutou e sanou minhas dúvidas – que não foram poucas – em todo esse período.

A todos da COPPE/UFRJ, meu obrigado por me receberem e me ajudarem desde o início. Aos professores do PESC, em especial aos professores Jano Moreira, Geraldo Xexéo e Marta Mattoso, obrigado por me aceitarem no programa e pelo conhecimento transmitido; à Cláudia Prata, Mercedes Souza, Mara Prata, Solange Santos, e todas estas pacientes mulheres que me guiaram pelos procedimentos e princípios da COPPE/UFRJ.

Aos meus companheiros no Laboratório Star One, que souberam aturar minhas dúvidas, ouvir conselhos e que me deram uma oportunidade única de crescer profissional e pessoalmente. Todos vocês são e serão lembrados como pessoas excelentes, trabalhadoras, competentes e obstinadas.

Ao CNPq pela bolsa de estudos que me auxiliou no desenvolvimento desta dissertação.

À minha família, saibam que chegou o momento de celebrar e receber aquilo que esperamos durante esse período. Acima de tudo, vejam que tudo, tudo valeu a pena. Nossa união, força e persistência. Não desistir do caminho nunca. Olhar sempre à frente e dar um passo de cada vez.

A minha namorada, que esse primeiro passo seja um dos alicerces da nossa vida a dois. Que nós possamos colher os frutos de todo esse esforço ao longo da nossa caminhada juntos. Ninguém, ninguém nos segura.

A todos os mestres que me trouxeram a calma, saúde e orientação nessa caminhada, meu agradecimento que ultrapassa as fronteiras da escrita e vai por todo coração, mente e alma. Assim como voçês me ajudam por toda a vida, que vocês também possam sempre continuar a crescer e olhar por todos nós.

Novamente a vocês meus orientadores, muito obrigado. Vocês sabem o como me senti agradecido, afortunado, feliz como uma criança pelos artigos publicados no *workshop* do VLDB e na revista indexada. Todo este resultado representou uma experiência única, está na memória como um grande incentivo para a vida como um todo.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

PROCESSAMENTO PARALELO DE CONSULTAS DE APOIO À DECISÃO EM GRIDS

Nelson Peixoto Kotowski Filho

Março/2008

Orientadores: Marta Lima de Queirós Mattoso
Alexandre de Assis Bento Lima
Programa: Engenharia de Sistemas e Computação

Consultas de apoio à decisão, materializadas comumente como consultas OLAP (*On-Line Analytical Processing*) são caracterizadas pelo elevado consumo de tempo e processamento e podem beneficiar-se de computação de alto desempenho. Esta dissertação apresenta a solução GParGRES, uma camada de software para gerenciar a execução de consultas de natureza tipicamente OLAP, de forma paralela e distribuída em ambientes de *grids* (grades). GParGRES é uma evolução da solução de *agrupamento* de banco de dados ParGRES: provê paralelismo inter - e intra-consulta, de forma não intrusiva aos SGBD e bases de dados pré-existentes e em um ambiente tipicamente heterogêneo e distribuído, a grade. A evolução de *agrupamento* de bancos de dados para grades computacionais apresenta desafios para o processamento paralelo de consultas OLAP. Este pode ser tratado em duas camadas de software; uma no nível de grade, com a distribuição de tarefas entre seus sítios e atenção especial ao balanceamento de carga e composição de resultados, e outra no nível de agrupamentos, com a redistribuição de tarefas entre seus nós. GParGRES foi concebido na forma de serviços para grade compatíveis com os padrões existentes e avaliado com base no *benchmark* TPC-H, que representa consultas OLAP, na plataforma *Grid'5000*, localizada na França.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

PARALLEL DECISION SUPPORT QUERY PROCESSING IN GRIDS

Nelson Peixoto Kotowski Filho

March/2008

Advisors: Marta Lima de Queirós Mattoso
Alexandre de Assis Bento Lima
Department: Computer Science and Systems Engineering

Decision support queries, usually denoted as OLAP (On-Line Analytical Processing) queries are typically time and CPU consuming and can take advantage of high performance computing. This dissertation presents GParGRES solution, a software layer in order to manage distributed and parallel OLAP query execution on grid environments. GParGRES is an evolution to the database cluster solution ParGRES: it provides inter and intra-query parallelism in a non intrusive DBMS and databases approach and inside a typically distributed and heterogeneous environment, the grid. The evolution from database clusters to computing grid states challenges to OLAP queries processing. It could be dealt within two software layers; one within grid level, with task distribution among sites and special care on load balancing and result composition, and another one at cluster level, with all appropriate task distribution through all its nodes. GParGRES was conceived according to existing grid services patterns, and has been implemented in Grid'5000, a grid platform initiative located in France and evaluated with the execution of TPC-H benchmark queries, which represent OLAP queries.

Índice do Texto

Índice de Figuras

Índice de Tabelas

Capítulo 1 – Introdução

1.1 – Motivação

Inicialmente desenvolvida para a comunidade científica, a computação em grades atrai maior interesse para outras áreas como sistemas EIS (Sistemas de Informação Executiva). A gestão de dados em grades (*grids*) começa a representar uma tarefa crítica, uma vez que deve ser possível garantir escalabilidade, ao mesmo tempo em que se deve endereçar fatores característicos de um ambiente de grade, como o dinamismo, heterogeneidade e autonomia das bases de dados envolvidas. (PACITTI, VALDURIEZ et al., 2007). IBM e Oracle, por exemplo, promovem ferramentas e serviços para grades corporativas (SHIMP e MIRANDA, 2005; WEBSPHERE, 2008).

A gestão de dados em grades foi abordada inicialmente através do uso de sistemas de arquivos distribuídos. De qualquer forma, soluções mais genéricas para bancos de dados são necessárias para tornar possível a virtualização de bancos de dados, que objetiva maximizar o uso dos recursos disponíveis, com plataformas e instâncias múltiplas e distribuídas de SGBD, ao mesmo tempo em que garante que o usuário não precisa possuir o conhecimento da localidade das bases de dados, SGBD e recursos. Dentre as possíveis formas de se prover tal funcionalidade, podemos citar a criação de serviços *web* e serviços para grade, por exemplo (AKBARINIA, PACITTI et al., 2007; ALPDEMIR, MUKHERJEE et al., 2003; BELL, BOSIO et al., 2002; SANTISTEBAN, GRAY et al., 2005; SMITH, GOUNARIS et al., 2002).

Em uma forma ideal, uma solução de banco de dados para grades deve respeitar a autonomia dos SGBD (evitar migração de aplicações e base de dados) e ao mesmo tempo beneficiar-se da computação paralela e distribuída fornecida pela grade. Para atender a tais pontos, pode-se desenvolver uma camada mediadora entre as aplicações externas ou internas a grade e os SGBD e bases de dados pertencentes a grade, que forneça processamento paralelo de consultas com uso de técnicas não intrusivas e enxergue os SGBD como componente caixa-preta.

Uma categoria importante de consultas em bancos de dados é representada pelas consultas de apoio à decisão, nesta dissertação também referenciadas como consultas OLAP (*Online Analytical Processing*). Estas tendem a acessar volumes extensos de dados e por consequência consomem elevado tempo para execução e demandam grande

capacidade de processamento. Geralmente, o responsável pela tomada de decisão tem prazo para concluir sua análise com base nestas consultas. Sendo assim, o processamento de consultas com alto desempenho é mandatário.

Uma solução típica para o processamento eficiente de consultas OLAP envolve o uso de técnicas que adotem o paralelismo inter e intra-consulta, através do uso de um SGBD paralelo em um sistema multi-processado ou em um agrupamento de PC (CECCHET, MARGUERITE et al., 2004; MATTOSO, ZIMBRÃO et al., 2005c; MATTOSO, ZIMBRÃO et al., 2005b; MATTOSO, ZIMBRÃO et al., 2005a; RÖHM, BÖHM et al., 2002). Por outro lado, esta solução requer tanto a migração em massa das bases de dados, processo caracterizado por ações de alto custo e complexidade, como a reavaliação do projeto físico e lógico das bases de dados, quanto à migração de aplicações existentes para o SGBD paralelo.

Uma alternativa que apresenta boa relação custo/benefício é representada pelo agrupamento (*cluster*) de bancos de dados, composto por um conjunto de PC, cada qual com seus processadores, discos e instalação de SGBD, como caixa-preta, interconectados por uma rede de alta velocidade (CECCHET, MARGUERITE et al., 2004; MATTOSO, ZIMBRÃO et al., 2005c; MATTOSO, ZIMBRÃO et al., 2005b; MATTOSO, ZIMBRÃO et al., 2005a; RÖHM, BÖHM et al., 2002).

Através da replicação parcial ou total da base de dados entre os nós do agrupamento, considerando cada SGBD como caixa-preta, o agrupamento de banco de dados pode ser utilizado por aplicações criadas com objetivo de prover paralelismo inter e intra-consulta de forma não intrusiva, através de, por exemplo, uma camada mediadora. O ParGRES (MATTOSO, ZIMBRÃO et al., 2005c; MATTOSO, ZIMBRÃO et al., 2005b; MATTOSO, ZIMBRÃO et al., 2005a) fornece o paralelismo inter e intra-consulta de forma transparente às aplicações que utilizem SGBD que suportem SQL-99 e tenham um controlador de acesso (*driver*) JDBC para conexões de cliente.

O processamento de consultas de apoio à decisão em ambientes de grades é alvo de pesquisa atual; existem iniciativas voltadas para nichos de atuação específicos e que demandam ambiente personalizado para execução, como no caso de soluções de armazéns de dados em grades (LAWRENCE, DEHNE et al., 2007; LAWRENCE e RAU-CHAPLIN, 2006; WERHLE, MIQUEL et al., 2005).

Alguns mediadores para grades também foram propostos, como no caso da OGSA-DAI, OGSA-DQP e Spitfire (ALPDEMIR, MUKHERJEE et al., 2003;

ANJOMSHOAA, ANTONIOLETTI et al., 2005; BELL, BOSIO et al., 2002; PATON, ATKINSON et al., 2002), mas estes não abordam ao mesmo tempo o paralelismo inter - e intra-consulta, o balanceamento de carga, a presença como serviço para grade e a característica de não-intrusão aos SGBD e transparência no processamento das consultas para a aplicação ou usuário final.

A migração de agrupamentos de PC para grades e o processamento de consultas de apoio à decisão em tais ambientes envolvem problemas que devem ser abordados em dois níveis: na grade, onde se deve tratar a distribuição de tarefas entre seus sítios, e nos agrupamentos, o que requer a redistribuição de tarefas entre cada máquina de cada agrupamento (considerando o cenário típico no qual cada sítio da grade é composto de um ou mais agrupamentos de PC).

O nível de grade requer atenção especial com relação ao balanceamento de carga, ao processamento das consultas e as técnicas de paralelismo adotadas, além de tratar a composição de resultados, operação responsável por coletar resultados provenientes da execução de cada sub-consulta executada em cada nó da grade e que foi gerada de acordo com a técnica de paralelismo adotada, em um único resultado final, correspondente ao esperado pela execução da consulta submetida originalmente. Todos estes processos devem levar em conta fatores como o cenário com alto dinamismo e heterogeneidade de *hardware*, aplicação (*software*) e fontes de dados disponíveis.

É neste sentido que propomos, nesta dissertação, o GParGRES (KOTOWSKI, LIMA et al., 2007; KOTOWSKI, LIMA et al., 2008), uma evolução da aplicação ParGRES (MATTOSO, ZIMBRÃO et al., 2005c; MATTOSO, ZIMBRÃO et al., 2005b; MATTOSO, ZIMBRÃO et al., 2005a). O GParGRES é uma camada de aplicação em forma de mediador, que abrange os tópicos acima citados, baseado em tecnologia e técnicas atuais de criação de serviços para grade, para o processamento de consultas de apoio à decisão em tais ambientes. GParGRES tem por base a aplicação ParGRES, que fornece processamento inter e intra-consulta de forma transparente no nível de agrupamentos de PC, enquanto o GParGRES atua no nível de nós da grade e trabalha com a replicação total da base de dados utilizada entre cada um dos mesmos.

O GParGRES fornece paralelismo inter e intra-consulta através de técnica de fragmentação virtual simples (AKAL, BÖHM et al., 2002) e as técnicas utilizadas para o desenvolvimento do mesmo caracterizam-se por seguirem os padrões atuais de criação de serviços para grade, por assegurarem a autonomia das bases dados e SBGD e seguem a filosofia de agrupamentos de bancos de dados.

GParGRES foi concebido de acordo com as técnicas atuais de desenvolvimento de serviços para grades, com base no arcabouço WSRF (GLOBUS WSRF, 2008), de forma que o GParGRES fornece operações distintas relacionadas ao conceito de serviços para grade, como a criação dinâmica de serviços e respectiva destruição dos mesmos, atribuição de estado ao serviço, entre outras.

No que diz respeito às atualizações de dados, estas são geralmente realizadas durante processos ETC (extração, transformação e carga), os quais ocorrem em janelas de tempo pré-determinadas, usualmente. Considera-se que ambientes OLAP empregam consultas tipicamente associadas à recuperação de dados, de forma frequente e que lidam com volume de dados extenso.

O GParGRES foi concebido e testado no *Grid'5000* (GRID'5000, 2008), uma plataforma de grade com alta escalabilidade, reconfigurável, localizada na França. Foram utilizados diferentes nós da grade em questão para a execução de experimentos e coletas de dados. Todos os experimentos foram realizados com base na execução de lotes de consultas que compõem o *benchmark* TPC-H (TPCH, 2008), projetado para representar aplicações OLAP.

Os resultados obtidos são listados e analisados nesta dissertação e apresentam indícios de que a mesma conseguiu endereçar os problemas listados e apresentou potencial de desempenho encorajador para a continuidade de uso e desenvolvimento de novas funcionalidades para o GParGRES.

1.2 – Organização do Documento

Esta dissertação foi organizada em cinco capítulos. O Capítulo 2 inicialmente apresenta a solução ParGRES e os conceitos por ela abordados, para que seja possível analisar o funcionamento de uma solução tradicional de agrupamento de bancos de dados, base do estudo para a solução proposta em grades, o GParGRES.

Ainda no Capítulo 2, apresentamos alguns dos mediadores e ambientes de grade existentes, com ênfase no ambiente adotado para realização de experimentos do GParGRES, o *Grid'5000*. Também são descritas as ferramentas de apoio à execução dos experimentos.

Em seguida, apresentamos os conceitos de serviços para grades, caminhos e padrões para desenvolvimento dos mesmos e algumas das iniciativas já existentes que abordam a utilização de bancos de dados em grades.

Finalmente, descrevemos o *benchmark* TPC-H, seu esquema da base de dados e consultas utilizadas na realização de experimentos do GParGRES.

O Capítulo 3 expõe o GParGRES, sua definição, arquitetura e processos, com os respectivos detalhes de implementação e características oferecidas pelo mesmo. Ao final deste capítulo, realizamos uma análise comparativa entre as funcionalidades oferecidas pelo GParGRES e as demais soluções analisadas nesta dissertação.

Em seguida, o Capítulo 4 apresenta a avaliação experimental do GParGRES. Para tal, é descrita a configuração do ambiente no qual o mesmo foi testado e os cenários utilizados. Este capítulo é encerrado com uma análise dos resultados obtidos.

Finalmente, as conclusões desta dissertação, assim como possíveis trabalhos futuros, são explicitadas no Capítulo 5.

Capítulo 2 – Bancos de Dados e Grades Computacionais

2.1 – ParGRES

A solução ParGRES é um mediador em forma de agrupamento de banco de dados que explora paralelismo inter e intra-consulta no processamento de consultas. O único requisito é que os SGBD adotados sejam compatíveis com o padrão SQL-99. O paralelismo é obtido através da replicação total de dados e da técnica de Fragmentação Virtual Adaptativa (AVP) (LIMA, MATTOSO et al., 2004). O ParGRES facilita a migração de bancos de dados provenientes de ambientes centralizados visto que não há requisito de mudanças no projeto físico do banco de dados associado. Ele provê flexibilidade com relação à alocação de nós para o processamento de consulta: quaisquer consultas podem ser processadas por qualquer conjunto de nós do agrupamento. A AVP fornece a base para o balanceamento dinâmico de carga entre os nós do agrupamento durante o processamento de consultas de forma não-intrusiva.

De forma similar a outros agrupamentos de banco de dados, o ParGRES gerencia a execução paralela de consultas através de instâncias de SGBD em cada nó do agrupamento. Produtos como C-JDBC (CECCHET, MARGUERITE et al., 2004) e PowerDB (RÖHM, BÖHM et al., 2002) utilizam uma camada de aplicação centralizada executada em um único nó, que atua como coordenador. Para evitar este gargalo, o ParGRES provê controle descentralizado uma vez que seus componentes são distribuídos entre os nós do agrupamento (Figura 1) (MATTOSO, ZIMBRÃO et al., 2005c; MATTOSO, ZIMBRÃO et al., 2005b; MATTOSO, ZIMBRÃO et al., 2005a).

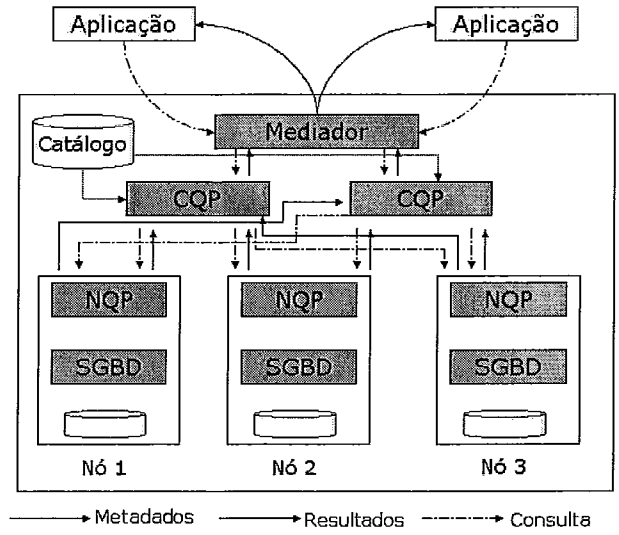

Figura 1 – Arquitetura do ParGRES

Fonte: (MATTOSO, ZIMBRÃO et al., 2005c)

Existem componentes locais e globais no ParGRES. Os componentes globais - mediador e CQP (*Cluster Query Processor*) - executam tarefas que envolvem diversos nós do agrupamento. Os componentes locais (SGBD e NQP (*Node Query Processor*)) executam tarefas relativas a um único nó.

O Mediador é responsável por receber as solicitações das aplicações, repassá-las ao CQP e realizar o retorno do CQP para as mesmas. O CQP coordena todos os demais componentes no contexto da consulta. Tendo em vista que a maioria dos agrupamentos possui um nó único acessível às aplicações externas (nó frontal), o Mediador é tipicamente alocado neste nó. Assim, apenas este componente é centralizado, o que traz flexibilidade na alocação física do CQP para cada requisição e aumenta a disponibilidade final do ambiente como um todo. Os componentes NQP coordenam localmente a execução da consulta no SGBD e conversam entre si durante o processo de distribuição de carga.

O ParGRES executa quatro tipos de tarefas:

(i) **Análise Sintática de Consultas SQL.** O CQP contém um analisador sintático que atua sobre os comandos SQL provenientes da aplicação cliente. Ele utiliza uma gramática livre de contexto para SQL-99. Os comandos que não sofrem análise por esta gramática são enviados diretamente ao SGBD. A informação gerada contém: (i) um conjunto de atributos e relações referenciadas pela consulta que pode ser utilizado para

obter paralelismo intra-consulta; (ii) informações necessárias para executar a composição de resultados; (iii) um conjunto de atributos utilizados nas operações de agregação.

(ii) Processamento de Consultas com Paralelismo Inter e Intra-Consulta. O CQP é responsável por escolher o tipo de paralelismo aplicado durante o processamento da consulta e a alocação dos nós que serão utilizados para tal. Ele utiliza informações do catálogo, que armazena metadados necessários para implementar a AVP apenas. Não há nenhuma informação específica com relação ao SGBD utilizado, o que preserva a autonomia em relação ao mesmo, considerando-o como um componente caixa-preta.

O paralelismo inter-consulta é relativamente imediato. O CQP envia a consulta ao nó NQP com o menor número de tarefas pendentes. A estratégia de paralelismo intra-consulta decompõe as consultas complexas em sub-consultas que serão executadas em paralelo sob diferentes fragmentos virtuais de dados. O CQP reescreve a consulta original em sub-consultas. Estas sub-consultas são uma versão da consulta original e contêm um predicado que determina os intervalos das partições virtuais, requeridos pela AVP. Estas são enviadas aos NQP, que realizam um ajuste fino específico dos fragmentos virtuais localmente. Cada NQP executa sua sub-consulta e gera um resultado parcial, que é então enviado ao CQP. Após receber todos os resultados parciais de todos os NQP, o CQP finaliza a composição do resultado e o envia à aplicação cliente.

A distribuição de dados não uniforme pode levar a desbalanceamento de carga. A técnica não intrusiva de balanceamento de carga do ParGRES aborda e resolve esta questão. Os NQP realizam o balanceamento através da troca de mensagens entre eles para redefinirem as partições virtuais. Os resultados em (LIMA, 2004) mostram que a técnica é eficiente, especialmente em casos de extremo desbalanceamento de carga.

(iii) Composição de Resultados. O ParGRES realiza a composição de resultados através de agregação em duas fases. Ele utiliza processamento paralelo nesta composição, o que minimiza a comunicação entre os nós. Na primeira fase, os nós agregam os grupos retornados pelas sub-consultas locais. Na segunda fase, os grupos são distribuídos aos seus respectivos nós através de uma função de mapeamento (*hash*). Por fim, cada nó envia seu subconjunto do resultado global ao nó coordenador, que realiza a sua união. Operações de ordenação também são realizadas em paralelo, de forma similar.

(iv) Processamento de Atualizações. Apesar de o ParGRES ter como ponto principal o processamento de consultas tipicamente OLAP para recuperação de dados, atualizações também podem ser realizadas. Tendo em vista que atualizações em ambientes OLAP são geralmente rápidas e executadas em momentos pré-determinados, o ParGRES adota uma política de consistência forte: ele não permite a execução de consultas de recuperação e atualizações ao mesmo tempo. Enquanto as atualizações são processadas, todas as demais consultas provenientes da aplicação do usuário final ficam bloqueadas. Quando existem somente consultas de recuperação de dados, o CQP permite a execução das mesmas em paralelo.

2.2 – Grades Computacionais e Serviços

Nesta seção, o conceito de grades é apresentado, assim como alguns dos ambientes de tal natureza existentes atualmente, cada qual com seu propósito e sob responsabilidade de entidades distintas.

2.2.1 – Definição

O conceito de grades foi apresentado à comunidade científica inicialmente como uma proposta de infra-estrutura computacional distribuída para oferecer suporte a processos de engenharia e ciências avançadas (FOSTER, KESSELMAN et al., 2001). Diferentemente de um agrupamento, bem como de uma rede ponto-a-ponto (P2P), o propósito de uma grade é prover um ambiente com poder computacional extenso e de forma colaborativa, de sorte que seja possível garantir aos seus associados o uso de recursos de acordo com suas respectivas necessidades e de forma compartilhada.

(FOSTER, KESSELMAN et al., 2001) então apresentam o conceito de organizações virtuais dinâmicas e escaláveis (*Virtual Organizations - VO*), que representa uma abstração para permitir que indivíduos, grupos de indivíduos e instituições em geral compartilhem recursos computacionais da forma controlada e colaborativa sugerida em sua definição do ambiente de grade.

A partir da criação de VO, é possível especificar a alocação de recursos físicos e de aplicação, como por exemplo, ciclos de máquina e funcionalidades de aplicativos, de forma organizada e distribuída, em relações de compartilhamento.

O conceito de VO pode ser também visualizado em (CARPENTER e JANSON, 2004), de forma similar ao citado por (FOSTER, KESSELMAN et al., 2001), como um

conjunto de recursos distribuídos e um respectivo conjunto de usuários que podem usufruir destes recursos, criando uma comunidade de grade, conforme retratado na Figura 2.

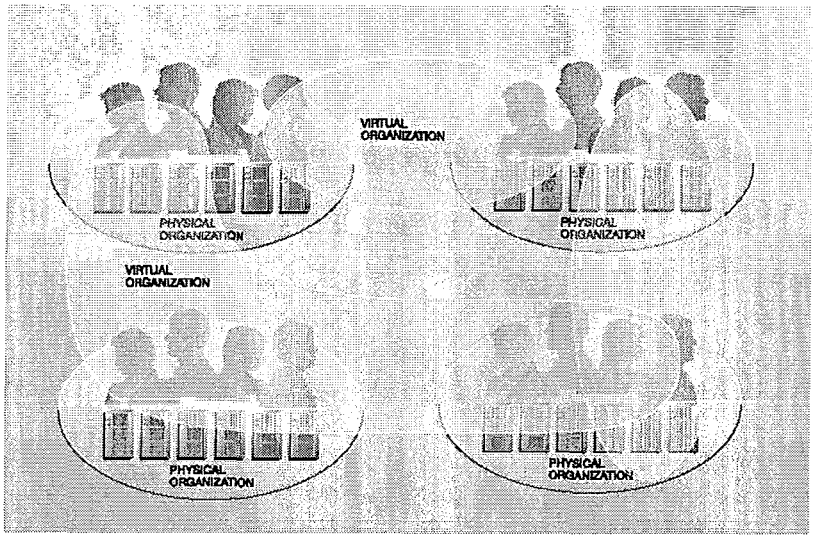

Figura 2 – Exemplo de Organizações Virtuais

Fonte: (CARPENTER e JANSON, 2004)

Atualmente, existem diversas iniciativas de instituições de vários países que têm por base o estabelecimento de ambientes de grades computacionais e, conseqüentemente, do desenvolvimento de aplicativos que possam ser executados em tal plataforma. Existem diversas áreas de pesquisa que podem beneficiar-se do potencial oferecido por uma grade, como bioinformática, física, entre outras.

2.2.2 – Globus

Iniciativa pioneira na área de grades, o *Globus* foi apresentado à comunidade científica em 1996 e atualmente é suportado pela *Globus Alliance* (GLOBUS, 2007), comunidade que inclui diversas universidades americanas e européias, bem como órgãos governamentais e empresas privadas, como IBM e Microsoft. O *Globus* representa uma das formas de mediadores de grade mais disseminadas ao redor do mundo.

Conforme visto em (FOSTER, 2005), o *Globus* é definido como:

1. Uma comunidade de usuários e desenvolvedores, que trabalham de forma colaborativa no desenvolvimento e uso de aplicação livre (código aberto) e sua respectiva documentação no contexto de computação distribuída e federação de recursos.
2. O *software Globus Toolkit*, atualmente em sua versão 4 (GT4), um conjunto de bibliotecas e aplicativos direcionados a problemas tradicionais relacionados à construção de serviços e aplicações distribuídas.
3. A infra-estrutura que oferece suporte a esta comunidade, através de repositórios de código, listas de correio eletrônico, entre outros.

Uma das características do *Globus* é permitir que seus usuários montem sua própria estrutura de grade ou se unam a grades pré-existentes, através do GT4. Para tal, é recomendado que o usuário tenha o conhecimento dos componentes necessários para o estabelecimento de uma grade com base no *Globus*. Ao mesmo tempo, o GT4 fornece a liberdade para o usuário desenhar sua infra-estrutura e implementá-la conforme desejar.

O GT4 oferece uma série de componentes e funcionalidades para viabilizar a construção e utilização de aplicativos e serviços em geral em um ambiente de grade. Seus conceitos principais estão descritos em (FOSTER, 2005) e são:

1. Segurança

A segurança padrão oferecida pelo GT4 é realizada através de protocolos e formatos de credenciais que têm por objetivo abordar a autorização, delegação, autenticação e proteção em geral das mensagens trafegadas na grade. É concretizada através de credenciais *X.509* isoladas ou em conjunto com identificadores e respectivas senhas.

2. Gerência de Dados

São três os pilares principais para a gerência de dados no GT4: a transferência de arquivos, através do GridFTP e da troca segura de arquivos (*Reliable File Transfer* - RFT); a replicação de dados e descoberta de serviços (*Replica Location Service* (RLS) e *Data Replication Service* (DRS); e o acesso a bancos de dados relacionais e/ou XML, através da arquitetura aberta de serviços para grade e integração de dados (*Open Grid Service Architecture – Data Access and Integration (OGSA-DAI)*).

3. Gerência de Execução

Através do aplicativo para gestão e alocação de recursos para grade (*Grid Resource Allocation and Management* (GRAM)), o GT4 oferece a capacidade de iniciar, monitorar e gerenciar a execução de operações arbitrárias nos recursos que compõem a grade.

4. Serviços de Informação – Monitoramento e Descoberta de Recursos (MDS)

O GT4 provê mecanismos para monitoramento e descoberta de recursos com base nas especificações WSRF e WSRF-*Notification*, embutidos em cada hospedeiro e serviço GT4 e com a função de interpretar as propriedades dos recursos com base XML.

5. Ambiente de Desenvolvimento e Execução

O GT4 oferece suporte para o desenvolvimento de hospedeiros de serviços *web* com base em linguagens Java, C e Phyton. Cada hospedeiro pode abrigar diferentes serviços, entre eles:

- Implementações da linguagem para descrição de serviços *web* (*Web Service Description Language* (*WSDL)*), protocolo de acesso a

objetos simples (*Simple Object Access Protocol (SOAP)*) e *WS-Security*, para funcionalidades básicas de serviços *web*.

- Especificações como *WSRF*, *WSRF-Notification* e *WS-Addressing*, para oferecer gerência de estado associado a serviços, recursos ou aplicações de forma geral.
- Hospedeiro Java para abrigar os diversos serviços *web* Java do GT4.
- Representação de informações de serviços em execução e monitoramento de sistema através de hospedeiros de recursos do tipo *WS-Resources*.

A Figura 3 apresenta a arquitetura do GT4 com maiores detalhes.

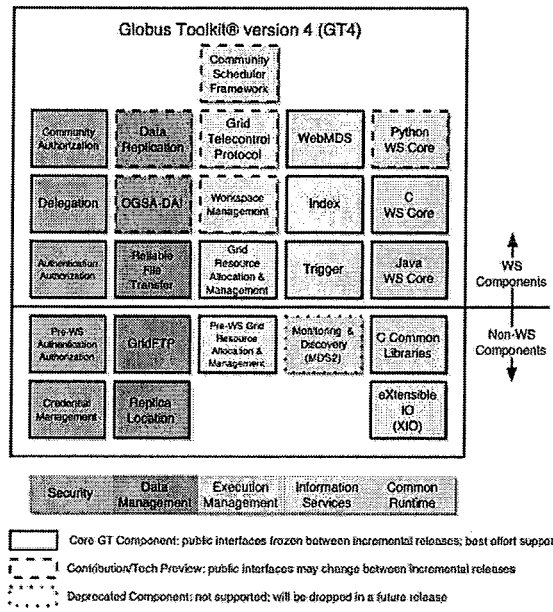

Figura 3 – Arquitetura do Globus Toolkit 4

Fonte: (GLOBUS, 2007)

Como se pode observar, a complexidade em montar e oferecer suporte para um ambiente com base no GT4 reside, entre outros fatores, no domínio do usuário sobre o mesmo.

2.2.3 – *Grid'5000*

Esta é uma iniciativa com base na França que tem por objetivo criar uma infra-estrutura em larga escala e reconfigurável de grade, com o propósito de fornecer um ambiente para realização de experimentos científicos. O seu nome está relacionado ao objetivo de alcançar a contagem de 5000 *CPUs.*

Atualmente, o *Grid'5000* (CAPPELO, DESPREZ et al., 2005) conta com nove sítios geograficamente distribuídos ao redor da França. Cada sítio do mesmo é um agrupamento propriamente dito, todos são interconectados entre si por uma rede de alta velocidade, de 1 a 10 Gbps, conforme apresentado na Figura 4.

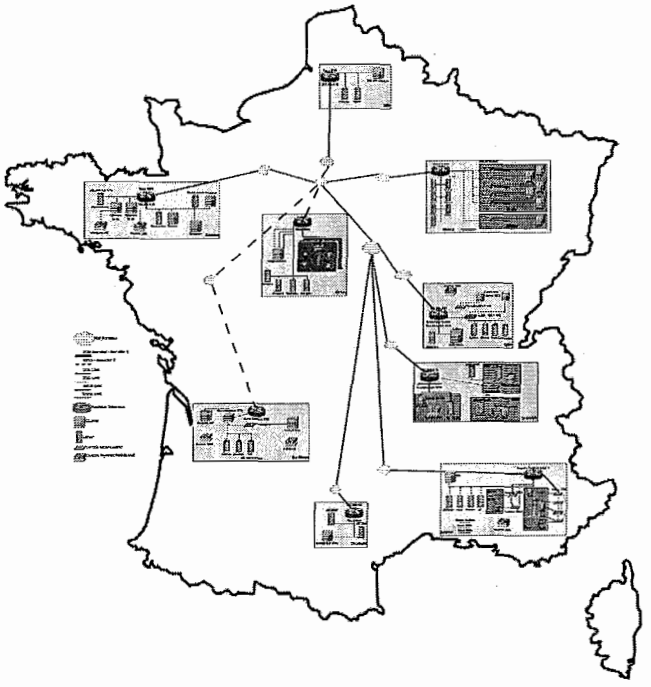

Figura 4 – Infra-Estrutura de Rede do *Grid'5000*

Fonte: (GRID'5000, 2008).

A Figura 5 apresenta a visão geográfica geral dos sítios que compreendem o *Grid'5000*.

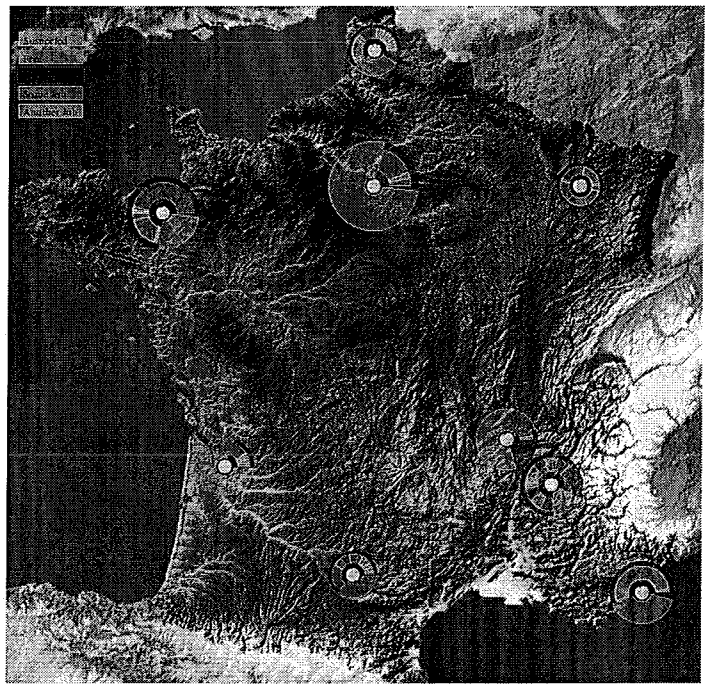

Figura 5 - Visão Geral do *Grid'5000*

Fonte: (GRID'5000, 2008).

A Tabela 1 apresenta a composição de processadores e núcleos de processamento de cada laboratório participante do *Grid'5000*.

A Tabela 2 retrata quantos são os nós disponíveis em cada sítio da plataforma.

A Tabela 3 apresenta a capacidade de disco oferecida pelo *Grid'5000*, por equipamento e sumarizada.

Tabela 1 – Processadores e Núcleos do *Grid'5000*

Fonte: (GRID'5000, 2008).

Sítio	Orsay	Grenoble	Lyon	Rennes	Sophia	Bordeaux	Lille	Nancy	Toulouse	Total
Processadores	684	270	260	524	356	424	198	334	276	**3326**
Núcleos	684	270	268	722	568	650	250	574	436	**4422**

Tabela 2 – Nós Disponíveis em cada sítio do *Grid'5000*

Fonte: (GRID'5000, 2008).

Sítio	Orsay	Grenoble	Lyon	Rennes	Sophia	Bordeaux	Lille	Nancy	Toulouse	Total
Processadores	342	135	130	260	178	202	99	167	137	**1650**

Tabela 3 – Capacidade de Disco do *Grid'5000*

Fonte: (GRID'5000, 2008) .

Hardware	Capacidade de Disco (GB)
Dell PowerEdge 1600SC	80
Dell PowerEdge 1855	70
Dell PowerEdge 1950	760
HP Integrity RX2600	72
HP ProLiant DL140G3	80
HP ProLiant DL145G2	80
HP ProLiant 385G2	72,8
IBM System x3455	80
IBM System x3755	600
IBM eServer 325	80
IBM eServer 326	80
IBM eServer 326m	80
Sun Fire V20z	73
Sun Fire X220 M2	250
Sun Fire X4100	146
Total Disponível	**2.603,8**

Segundo (CAPPELO, DESPREZ et al., 2005), o projeto do *Grid'5000* segue os princípios de:

1. Facilidade de Reconfiguração dos Nós

Através de um aplicativo próprio denominado Kadeploy2 (KADEPLOY2, 2008), é possível que cada usuário da grade crie seu ambiente de sistema em uma partição de disco pré-determinada automaticamente. Por ambiente de sistema entende-se toda a gama de aplicativos, desde o sistema

operacional a ser utilizado até os programas a serem executados para os experimentos científicos.

O usuário gera um nome para o ambiente a ser criado, os nós são reiniciados via rede, os discos são preparados e demais configurações realizadas até que os nós sejam finalmente reiniciados com a imagem do sistema desejado, também via rede.

2. Segurança em Dois Níveis

Nenhuma de suas máquinas está diretamente conectada à internet. Usuários conectam-se ao *Grid'5000* através de nós frontais existentes em cada sítio da plataforma. Este nó atua como porta de entrada e fornece mecanismos de segurança e controle de acesso, como um intermediário aos demais agrupamentos e nós do *Grid'5000*. Pode ser necessário o uso de aplicativos clientes de conexões virtuais privadas em sítios que assim o requeiram (VPN). Todo pacote de comunicação entre sítios é transmitido sem limitações.

O propósito do *Grid'5000* é beneficiar a execução de experimentos na grade. Mecanismos mais sofisticados de segurança, como a utilização de infra-estruturas de chaves públicas e certificados *X.509*, já adotados em iniciativas como o *Globus* (FOSTER, 2005), geram carga adicional de processamento e impactam diretamente no desempenho da solução. Sendo assim, a solução de segurança adotada pelo *Grid'5000* contempla o uso de redes virtuais privadas do tipo MPLS (MPLS, 2008), que trabalham em conjunto com o protocolo IP para garantir isolamento de tráfego e redirecionamento de pacotes e para minimizar a sobrecarga na interconexão entre os sítios, com base em tecnologias fornecidas pelo RENATER (RENATER, 2007).

Já entre as máquinas da grade, foi adotada a utilização de tecnologia *LDAP*, onde cada sítio tem um servidor *LDAP* com árvores de estrutura idêntica. Ressalta-se que os dados de cada usuário são locais para cada sítio. Apesar de serem compartilhados em cada agrupamento via *NFS*, a distribuição entre sítios é de responsabilidade do próprio usuário, através de ferramentas como *SCP*, *SFTP* entre outras.

3. Homogeneidade dos Nós

Para garantir melhor qualidade no suporte de infra-estrutura, facilitar o processo de reconfiguração dos nós e diminuir custos de manutenção, o *Grid'5000* optou por conservar 2/3 de seu ambiente de forma homogênea, no que diz respeito aos recursos físicos (*hardware*) utilizados. Ainda assim, segundo os criadores da plataforma, a mesma atende aos requisitos de heterogeneidade usualmente propostos a um ambiente de grade.

4. Gerenciamento de Recursos e Execução de Tarefas

O *Grid'5000* oferece aos seus usuários uma ferramenta denominada OAR2 (OAR2, 2008), que trabalha tanto no nível dos agrupamentos, como um sistema de gestão de recursos, quanto no nível da grade como um todo, com o papel de gerenciador e agendador de tarefas submetidas a grade.

Esta ferramenta permite que experimentos sejam agendados através de linha de comando, a partir dos nós frontais de cada sítio. Sendo assim, é possível especificar janelas de tempo desejadas que satisfaçam tanto à necessidade do usuário quanto à disponibilidade de recursos da grade. Também é possível solicitar, a critério do usuário, quantos e quais nós serão utilizados em um experimento. Além disto, cada reserva pode ser caracterizada como interativa ou não, neste último caso associada à execução de rotinas pré-programadas.

Sendo assim, o aplicativo OAR2 provê a reserva dos nós, enquanto o Kadeploy2 fornece a imagem de sistema operacional personalizada para a execução de cada experimento do usuário. Esta abordagem é interessante para assegurar a exclusividade de uso da grade para um solicitante em detrimento dos demais usuários dos sítios do mesmo, caso seja preciso, por exemplo, realizar uma avaliação de um *benchmark* aplicado a uma ferramenta, como no caso do GParGRES.

5. Controle e Avaliação de Resultados

Através da combinação de todas as funcionalidades oferecidas pelo *Grid'5000*, o mesmo passa a ter a característica de um ambiente de grade para experimentos científicos com maior controle. Além disso, este incorpora as facilidades de gerência e reconfiguração de nós da grade e provê segurança de forma a não prejudicar o desempenho dos aplicativos a ele submetidos.

Sendo assim, o *Grid'5000* pode ser considerado como uma infra-estrutura de grade que oferece um bom nível de homogeneidade entre seus nós, ao mesmo tempo em que não elimina a heterogeneidade típica de uma grade. Além disso, este oferece uma ferramenta para facilitar a reconfiguração de tais nós, ao mesmo tempo em que fornece segurança ao ambiente e aplicativos que auxiliam na personalização de ambientes, gestão de recursos e submissão de tarefas.

Conforme pode ser observado em (CAPPELO, DESPREZ et al., 2005), o *Grid'5000* suporta inclusive a instalação do mediador *Globus Toolkit* e assim permite a criação de uma grade virtual interna, capaz de executar procedimentos submetidos de forma similar ao uso do *Globus* isoladamente. Dessa forma, agregam-se as vantagens tanto do *Globus* quanto do *Grid'5000* para o usuário final e o próprio desenvolvedor.

2.2.3.1 – Escalonamento de Tarefas e Gerência de Recursos

Com a função de gerenciar recursos e atuar como um agendador de tarefas no *Grid'5000*, o OAR2 (OAR2, 2008) permite aos usuários configurar reservas de nós em agrupamentos da grade com o uso de diversos parâmetros, que podem ser utilizados isoladamente ou em conjunto.

Dentre os parâmetros mais relevantes para esta dissertação, destacam-se a possibilidade de selecionar o agrupamento desejado, a quantidade de nós desejada, a janela de tempo (em horas) para a reserva, data de início e modo de funcionamento da reserva (interativa, com acesso via protocolo ssh, ou sequencial, para execução de rotinas pré-programadas).

A solicitação de janelas de tempo para execução de experimentos através da reserva dos nós é realizada através da linha de comando, a partir de cada nó frontal de cada agrupamento do *Grid'5000*. O OAR2 trata automaticamente conflitos relativos aos parâmetros solicitados e informa o usuário sobre a o sucesso ou não da reserva, com um identificador numérico associado à mesma. Uma vez efetuada a reserva de nós com sucesso, é possível acompanhar o andamento da tarefa gerada através de uma interface gráfica *web*, disponível no sítio do *Grid'5000* e apresentada na Figura 6.

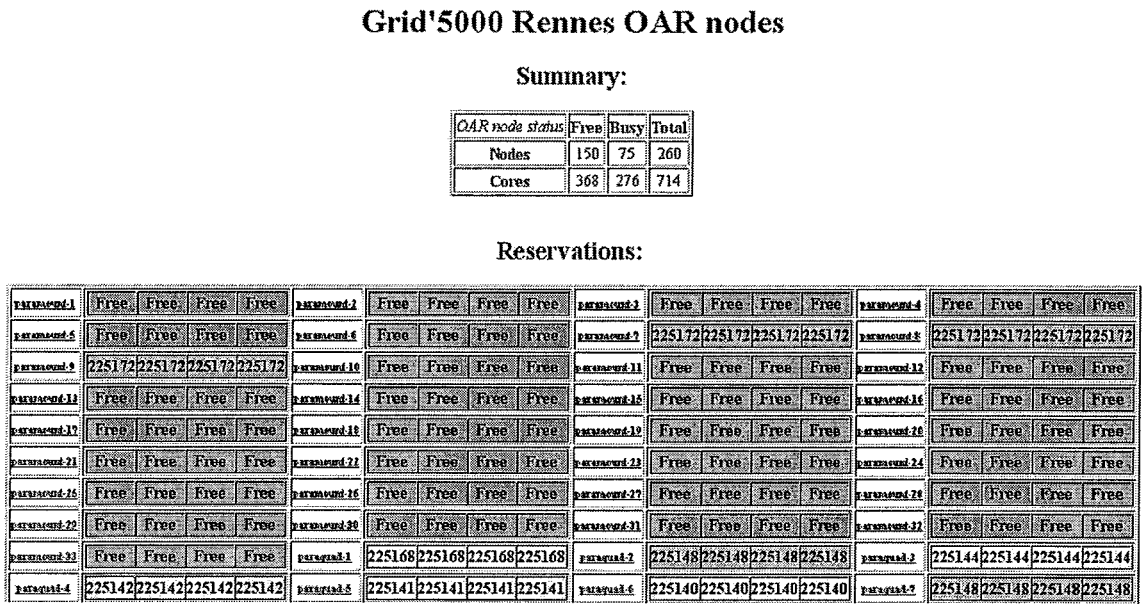

Figura 6 – Diagrama das reserva de nós via OAR2

A Figura 7 demonstra o detalhamento de cada reserva, com informações referentes ao usuário, número de nós, janela de tempo, entre outros critérios adotados pela reserva em questão.

Job details:

Id	User	State	Queue	NbNodes	NbCores	Type	Properties	Reservation	Walltime	Submission Time	Start Time	Scheduled Start
224900	pmorillo	Waiting	default	ALL	0	INTERACTIVE	deploy = 'YES'	Scheduled	4:0:0	2008-02-15 10:32:58	2008-02-21 08:00:00	2008-02-21 08:00:00
225127	lsteffenel	Waiting	default	40	0	INTERACTIVE	deploy = 'YES'	Scheduled	8:0:0	2008-02-18 12:02:10	2008-02-19 10:00:00	2008-02-19 10:00:00
225140	ethome	Running	besteffort	8	32	PASSIVE	besteffort = 'YES'	None	48:0:0	2008-02-18 13:21:45	2008-02-18 13:21:46	2008-02-18 13:21:46
225141	ethome	Running	besteffort	8	32	PASSIVE	besteffort = 'YES'	None	48:0:0	2008-02-18 13:21:46	2008-02-18 13:21:48	2008-02-18 13:21:48
225142	ethome	Running	besteffort	8	32	PASSIVE	besteffort = 'YES'	None	48:0:0	2008-02-18 13:21:47	2008-02-18 13:21:48	2008-02-18 13:21:48
225143	ethome	Running	besteffort	8	32	PASSIVE	besteffort = 'YES'	None	48:0:0	2008-02-18 13:21:47	2008-02-18 13:21:51	2008-02-18 13:21:51
225144	ethome	Running	besteffort	4	16	PASSIVE	besteffort = 'YES'	None	48:0:0	2008-02-18 13:21:49	2008-02-18 13:21:51	2008-02-18 13:21:51
225145	ethome	Running	besteffort	4	16	PASSIVE	besteffort = 'YES'	None	48:0:0	2008-02-18 13:21:50	2008-02-18 13:21:51	2008-02-18 13:21:51
225148	dancelin	Running	default	12	48	INTERACTIVE	deploy = 'YES'	Scheduled	7:59:25	2008-02-18 14:16:31	2008-02-18 14:16:33	2008-02-18 14:16:33
225149	dancelin	Running	default	6	24	INTERACTIVE	deploy = 'YES'	Scheduled	7:59:25	2008-02-18 14:18:01	2008-02-18 14:18:03	2008-02-18 14:18:03
225161	jccharr	Waiting	default	150	0	INTERACTIVE		Scheduled	5:0:0	2008-02-18 16:08:25	2008-02-19 14:00:00	2008-02-19 14:00:00
225168	ethome	Running	besteffort	14	32	PASSIVE	besteffort = 'YES'	None	48:0:0	2008-02-18 19:30:43	2008-02-18 19:30:44	2008-02-18 19:30:44
225170	rpaescam	Waiting	default	16	0	INTERACTIVE	(cluster="paraquad") AND deploy = 'YES'	Scheduled	6:0:0	2008-02-18 20:06:27	2008-02-20 17:00:00	2008-02-20 17:00:00
225172	npeixoto	Running	default	3	12	INTERACTIVE	(cluster="paramount") AND deploy = 'YES'	Scheduled	5:44:23	2008-02-18 22:15:35	2008-02-18 22:15:37	2008-02-18 22:00:00

Figura 7 – Detalhes das tarefas em execução no OAR2

2.2.3.2 – Personalização de Ambiente de Sistema

Assim como o *Grid'5000* oferece ferramentas que auxiliam na gestão de recursos e no agendamento de tarefas, existe o aplicativo Kadeploy2, um "sistema para implantação, escalável e rápido, direcionado à computação em agrupamentos e grades" (KADEPLOY2, 2008). O objetivo do mesmo é facilitar a criação de ambientes de sistema operacional e demais aplicativos (SGBD, bases de dados, servidores *web*, aplicações do usuário, entre outras) em uma imagem, na forma de um arquivo que pode ser carregado e implantado em um ou mais nós computacionais.

No *Grid'5000*, são disponibilizadas imagens de sistema operacional Linux denominadas base, para cada nó frontal de cada sítio, que podem ser utilizadas como referência para criação de imagens personalizadas. Através de tal imagem e uma reserva

de nó via OAR2, um usuário pode iniciar um nó com a mesma e uma vez carregada, instalar demais aplicativos que ache necessário. Uma vez finalizada a personalização do ambiente, o usuário então cria uma imagem do sistema na forma de um arquivo .tgz, salva-a em local apropriado e associa o mesmo a uma imagem diretamente relacionada a seu identificador de usuário no Kadeploy2.

A aplicação é utilizada totalmente via linha de comando, sem o apoio direto de uma interface, mas sua simplicidade de uso e documentação facilitam a operação do usuário. Existem outras operações, como o reinício de nós, a exclusão de imagens, o arquivamento de imagens de usuários, entre outros, que também podem ser úteis de acordo com o experimento a realizar.

A Figura 8 apresenta todo o processo de carregamento de uma imagem personalizada (gpargres) em um nó (paramount-8.rennes.grid5000.fr), através do uso do Kadeploy2.

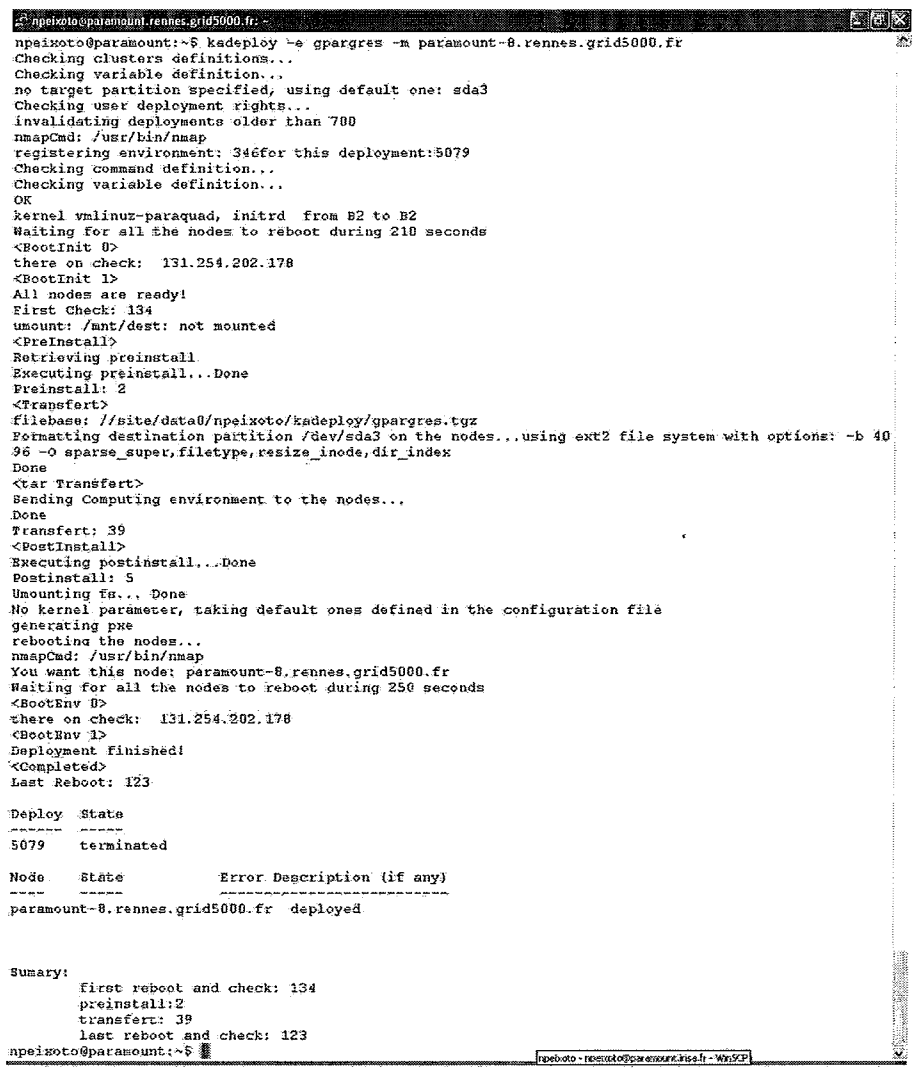

Figura 8 – Exemplo de uso do Kadeploy2

2.2.4 – EELA

O EELA (*E-Infrastructure Shared Between Europe and Latin America*) (EELA, 2007) representa um projeto de cooperação entre países da Europa (Espanha, Portugal e Itália) e América Latina (Argentina, Brasil, Chile, Cuba, México, Peru e Venezuela), na área de grades computacionais. O EELA possui três objetivos em nível macro, especificados em (EELA, 2007):

1. Estabelecer uma rede humana de colaboração.
2. Criar um piloto de infra-estrutura de grade na América Latina, interoperável para com o EGEE (EGEE, 2007) (*Enabling Grids for E-Science*), existente na Europa, como uma forma de transferência de conhecimento e suportando a execução de aplicativos relevantes.
3. Identificar e promover um arcabouço sustentável para o *e-Science*.

O EELA conta com a participação de diversas universidades, entre elas a UFRJ, no sentido de promover a montagem e uso do ambiente de grade desejado e sua integração ao EGEE, que atualmente encontra-se em sua segunda fase, sendo conhecido também como EGEE-II. A Figura 9 apresenta a visão geral do EELA.

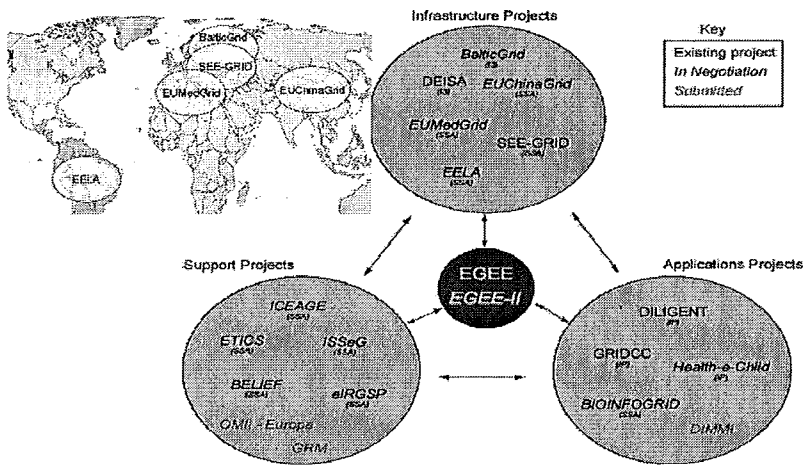

Figura 9 – Interface entre EELA, EGEE (EGEE-II) e demais grades

Fonte: (EELA, 2007)

A base para o desenvolvimento do EELA é estabelecida pelo uso da ferramenta Globus, assim como do mediador gLite (GLITE, 2007), base para a construção do EGEE.

Existem quatro áreas de *e-Science* que são referenciadas pelo EELA como relevantes para o projeto: aplicações biomédicas, física de alta energia, *e-learning* e meteorologia.

Outro ponto relevante do EELA está no fato de, inicialmente, sua proposta fomentar explicitamente o uso dos recursos e aplicações já desenvolvidas, em operação e suportadas pelo EELA/EGEE. Conforme pode ser observado em (EGEE, 2007), "Adaptações mínimas poderão ser consideradas somente para garantir a interoperabilidade entre as iniciativas de grade Européias e Latino-americanas, em particular para com o EGEE".

2.3 – Serviços para Grade e Padrões de Desenvolvimento

2.3.1 – Introdução

Nesta seção demonstramos a definição dos serviços para grade como evolução dos serviços web e apresentamos um arcabouço genérico para construção de aplicações que concretizem as operações de um serviço para grade. Finalmente, uma implementação concreta deste arcabouço, escolhida para o desenvolvimento do GParGRES é apresentada em detalhes.

Para tal, inicialmente definimos o conceito primitivo que permitiu a concepção dos serviços para grade, os serviços *web*.

2.3.2 – Serviços *Web*

Conforme definição apresentada pelo W3C (W3C, 2007), um serviço *web* é um sistema projetado para suportar a interação máquina-máquina sob uma rede (BOOTH, HAAS et al., 2004). Os serviços *web* contam com uma interface projetada com base na linguagem WSDL (WSDL, 2008) e a interação entre aplicativos ocorre através da troca de mensagens no protocolo SOAP (SOAP, 2008) via arquivos XML trafegados por protocolo HTTP.

Os serviços *web* definem uma técnica para descrever componentes de aplicação a serem acessados, métodos para acessá-los e métodos de descoberta que possibilitem a identificação de fornecedores de serviços (FOSTER, KESSELMAN et al., 2002).

Outra abordagem objetiva para a definição de serviços *web* é a de que estes são tecnologias que fornecem um modelo de programação independente de ambiente e linguagem de desenvolvimento, o que acelera a integração de aplicativos dentro e fora de uma organização. Estas tecnologias contam com interfaces que descrevem o conjunto de operações passíveis de execução em um ambiente de rede através da troca de mensagens XML (GOTTSCHALK, GRAHAM et al., 2002; SOTOMAYOR, 2005).

Outro fator relevante para a compreensão do conceito de serviços *web* é o de que estes são distintos dos navegadores *web* e tecnologia HTML. Apesar de serem construídos com base no protocolo HTTP, os serviços *web* não são originalmente projetados para serem utilizados por usuários finais tradicionais, mas sim entre aplicativos (SOTOMAYOR, 2005).

Conforme citado, os serviços *web* utilizam diversas tecnologias para sua implementação e uso. Três destas tecnologias foram projetadas especificamente para os serviços *web*, diferentemente do protocolo HTTP e do padrão XML. Estas são definidas em (FOSTER, KESSELMAN et al., 2002) e podem ser sumarizadas como:

1. Protocolo de acesso a objetos simples (SOAP) – protocolo para troca de mensagens entre o solicitante e o fornecedor do serviço propriamente dito.
2. Linguagem para descrição de serviços *web* (WSDL) – linguagem utilizada para criação de um documento (em formato similar ao da linguagem XML), que descreve um serviço *web*, através de um conjunto de tipos de dados, mensagens, operações, falhas previstas e portas (*endpoints*) que operam sobre chamadas RPC ou através de mensagens via documentos (ou arquivos).
3. WS-Inspection – além de um documento em linguagem WSDL, existe outra alternativa para localizar descrições de serviços fornecidos, através de uma entrada de registro UDDI (*Universal Description, Discovery and Integration*).

A Figura 10 apresenta um exemplo de execução, desde a invocação até a resposta ao usuário final, de um serviço *web*, de forma simplificada.

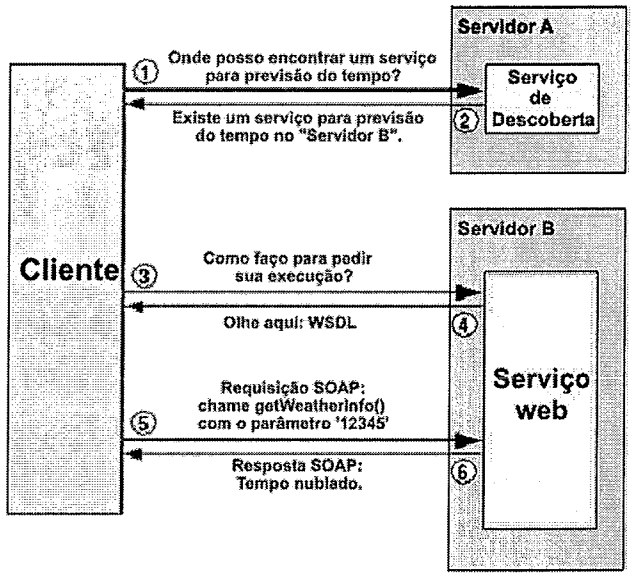

Figura 10 – Serviço *web* em Execução

Fonte: (SOTOMAYOR, 2005).

Portanto, (SOTOMAYOR, 2005) observa que serviços *web* tendem a ser adequados para sistemas com baixo acoplamento e que operem em escala de Internet, nos quais o usuário não tem conhecimento prévio sobre o componente em si, mas será capaz de identificar dentre as operações fornecidas pelo serviço *web* quais as que podem satisfazer suas necessidades (SOTOMAYOR, 2005).

Pode-se então indagar sobre a possibilidade de uso de serviços *web* em grades. Dentre as vantagens apresentadas pelos serviços *web* para o contexto de grades, podemos citar a necessidade de compor e oferecer serviços em ambientes heterogêneos, através de interfaces padronizadas e independentes de plataforma ou linguagem utilizada. Além disto, a adoção em larga escala de serviços *web* proporciona maior número de ferramentas e funcionalidades desenvolvidas e postas à disposição da comunidade científica (FOSTER, KESSELMAN et al., 2002).

Por outro lado, é possível apontar algumas desvantagens dos serviços *web*, em geral:

1. Sobrecarga - a transmissão de dados via XML pode ocasionar desempenho abaixo do esperado em comparação ao uso de código binário tradicional.

2. Dinamicidade e Ciclo de Vida – em serviços *web*, não contamos com dinamismo, no sentido em que uma instância de serviço deve permanecer ativa para que um cliente solicite a realização de uma determinada operação por ele prestada. Dessa forma, não a gestão de ciclo de vida de um serviço fica restrita a mantê-lo operacional e ativo para que seja possível atender às requisições provenientes de outras aplicações, impossibilitando a criação dinâmica de uma instância de serviço.

3. Persistência – serviços *web* não carregam consigo a possibilidade de utilização de objetos de propriedade, relacionados ao estado de uma instância de serviço, por exemplo.

Sendo assim, serviços *web* representam uma alternativa para exposição de operações de forma que as demais aplicações não necessitem conhecer o funcionamento interno ou realizar chamadas específicas para o serviço desejado, ou seja, existe o baixo acoplamento, o que representa como um ponto positivo para ambientes heterogêneos e em larga escala.

Ao mesmo tempo, existem pontos a desenvolver, como a gestão de ciclo de vida, dinamismo e persistência dos serviços, por exemplo. A seguir apresentamos a definição de serviços para grade, que buscam atender a tais questões.

2.3.3 – Serviços para Grade

Serviços para grade são apresentados em (FOSTER, KESSELMAN et al., 2002) como uma evolução do conceito de serviço *web*, capazes de endereçar as seguintes particularidades:

1. Criação Dinâmica do Serviço - este pode ser instanciado em tempo real para cada cliente, ou seja, não é necessário que este já esteja em execução antes de ser solicitado ou permanentemente.

2. Evocação confiável e segura - a confiabilidade na evocação e execução de serviços para grade está relacionada em assegurar que mensagens foram entregues de um serviço a outro, através do conceito inerente de estado para cada um dos mesmos. Além disso, mecanismos de autenticação e protocolos de seguranças podem ser oferecidos para reforçar a implantação de políticas de segurança e minimizar a possibilidade de problemas relacionados a este fator.

3. Gestão do Ciclo de Vida do Serviço - deve ser possível estabelecer limites de tempo para sua execução e, desta forma, intervir para sua finalização e conseqüente liberação de recursos.

4. Notificação - uma coleção de serviços dinâmicos deve ter a capacidade de se comunicar internamente, para que seja possível informarem uns aos outros sobre mudanças relevantes de estado que demandem alguma ação ou disparem algum evento específico.

5. Virtualização - abstração que permite a criação de serviços mais complexos, refinados, a partir da composição de outros, sem que haja a preocupação em como estes foram implementados ou compostos.

Para alcançar estes objetivos, foi então estabelecida a arquitetura denominada OGSA (*Open Grid Service Architecture*), apresentada em (FOSTER, KESSELMAN et al., 2002). Esta utiliza a WSDL como forma para definir os padrões de interfaces dos serviços, que, por sua vez, passam a conter a informação de estado internamente.

A incorporação do conceito de estado é referenciada como o fator que diferencia serviços para grade dos serviços *web* em (SOTOMAYOR, 2005). Apesar de não existirem restrições na arquitetura destes que excluam a noção de estado, reforça-se que os denominados serviços *web* não são desenvolvidos de forma a terem uma memória própria, que os torne capazes de conservar estados entre uma execução e outra.

Através destas abordagens, pode-se concluir que um serviço para grade é um conceito em evolução, mas já existem componentes comuns relacionados ao mesmo, como a noção de estado. Além disso, estes são desenvolvidos para suportar a execução de aplicações em ambientes com características dinâmicas.

Uma vez estabelecido o conceito de serviços para grade, é preciso estudar o arcabouço que estabelece o primeiro passo na concretização de serviços de tal natureza,

que é representado pelo arcabouço *Web Services Resource Framework* (WSRF, 2008), apresentado a seguir.

2.3.3.1 – Web Services Resource Framework (WSRF)

O objetivo do WSRF é "definir um arcabouço genérico para modelar e acessar recursos persistentes através de serviços *web*, de tal forma que a definição e implementação de um serviço e a integração e gerência de serviços múltiplos seja facilitada" (WSRF, 2008). Esta definição atende aos requisitos apontados pelo GParGRES, pois faz referência ao conceito de estado de um serviço (persistência), à como criar e atribuir propriedades persistentes a serviços *web*, que então passariam a ser caracterizados como serviços para grades.

O WSRF introduz um documento em formato XML para a descrição das propriedades de um recurso (*Resource Properties Document*), que é referenciado pelo arquivo WSDL que descreve o serviço e expõe suas características para o cliente final. Dessa forma, passa-se a ter o conceito de um recurso do tipo WS-*Resource*, no qual as requisições são simplificadas, já que o identificador do objeto a ser manipulado por uma requisição SOAP não precisa ser incluído no corpo de cada ação desejada; apenas é necessário informá-lo no cabeçalho do envelope SOAP. A Figura 11 demonstra as diferenças entre envelopes SOAP de serviços não WSRF (à esquerda) e serviços tipicamente WSRF (à direita).

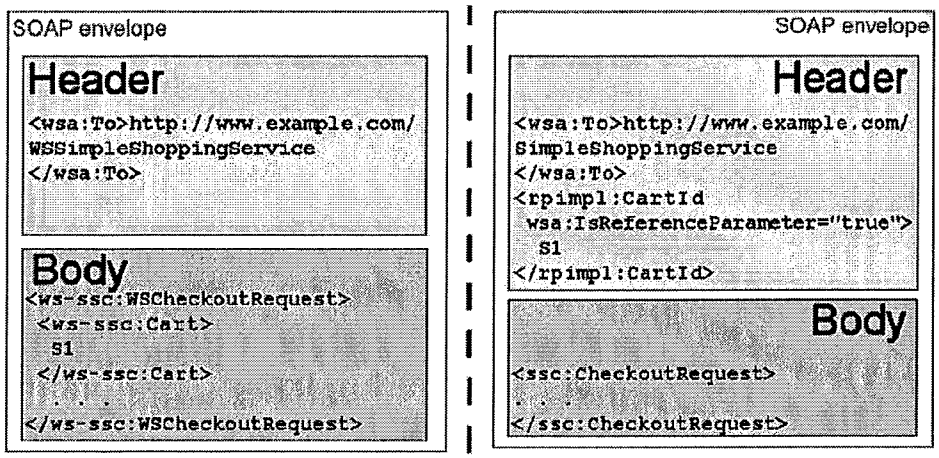

Figura 11 – Mensagens SOAP entre serviços não-WSRF vs. WSRF

Fonte: (SOTOMAYOR, 2005; WSRF, 2008).

Com a adoção do WSRF, o próprio cabeçalho do documento de propriedades passa a concentrar informações como o identificador da instância do serviço referenciada no momento, o estado da mesma ou demais características projetadas para o serviço em questão. Assim, além de simplificar a referência a uma instância de recurso *web*, o WSRF padroniza as mensagens necessárias para recuperar ou substituir uma ou todas as propriedades listadas no documento de propriedades do mesmo.

O WSRF também padroniza outras questões relacionadas aos recursos WS-*Resources*, que incluem: gestão e controle do ciclo de vida de um recurso (WS-*ResourceLifetime*); gestão de falhas de um recurso (WS-*BaseFaults*); agregação de informações sobre recursos e serviços (WS-*ServiceGroup*); descrição de mensagens para notificações de mudanças nos estados dos recursos (WS-*BaseNotification*).

Cada um destes padrões, suas operações e atributos são mapeados através de arquivos WSDL e respectivos esquemas XSD (XSD, 2008), arquivos em linguagem XML que estabelecem padrões para definições de estruturas e conteúdo de documentos em XML, quando aplicáveis. Estes padrões não estão associados a nenhuma linguagem de programação específica. A implementação concreta, para uso em uma aplicação, deve ser realizada por quem interessar, com uso dos arquivos WSDL e XSD respectivos como padronização de acesso aos recursos desejados.

Como exemplos, temos as implementações do projeto Apache Muse (MUSE, 2008), Globus Toolkit 4 (GLOBUS WSRF, 2008) e IBM WSRF for WebSphere (WEBSPHERE, 2008).

As operações definidas pelo conjunto de documentos WSDL oferecidos pelo WSRF interessam ao GParGRES pois endereçam questões como: estado de um serviço, que pode ser mapeado como uma propriedade em um documento de propriedades; gestão de ciclo de vida, para lidar com questões como auxiliar na destruição de um recurso após seu uso, caso desejado, e na concepção de um tempo limite de execução; gestão de falhas, para decidir sobre quais ações tomar em caso de problemas encontrados durante a execução de ações relacionadas ao mesmo; agrupamento de instâncias de serviços, para atender a um ou mais usuários finais; e a padronização da troca de mensagens entre objetos que componham a solução GParGRES.

Como o objetivo da solução GParGRES é oferecer suporte à execução de consultas tipicamente OLAP em ambientes de grades, através de serviços que possam ser aderentes ao conceito WSRF, foi preciso pesquisar e analisar quais das implementações do arcabouço em questão, disponíveis na comunidade, poderiam servir

de base para o desenvolvimento do GParGRES e atender aos princípios de não-intervenção/dependência de mediador para a grade e uso de uma plataforma de código aberto.

A seguir, apresentamos a implementação escolhida e pontos que motivaram tal decisão.

2.3.3.2 – Apache Muse

O Apache Muse é "uma implementação das especificações do arcabouço WSRF, WS-*BaseNotification* (WSN) e WS-*DistributedManagement* (WSDM)" (MUSE, 2008). Através dele, é possível construir recursos no padrão WS-*Resource* e usar as operações previstas pelos padrões citados através do uso conjunto de bibliotecas e arquivos WSDL e XML oferecidos pelo Muse.

O Apache Muse expõe cada funcionalidade desejada pelo desenvolvedor do serviço projetado como uma operação capacitadora (*capability*), que representa uma operação chave de um serviço, enquanto que os demais métodos internos às classes do mesmo continuam como métodos Java tradicionais, não visíveis para aplicações externas. Cada operação capacitadora pode ser alocada a um recurso (*resource*), que equivale a um conjunto das mesmas. Finalmente, cada serviço pode fazer referência a um conjunto de recursos, cada qual com suas operações externas. Através da definição de um arquivo WSDL para o serviço a ser exposto, é gerado um arquivo em formato XML que identifica todos os recursos, agrupa suas respectivas operações externas e lista as classes Java que as implementam.

Um recurso, depois de desenvolvido, é então empacotado em um arquivo WAR (*Web Application Archive*), uma extensão de um arquivo JAR (*Java Archive*), voltado para aplicações *web* em Java, passível de implantação no Apache Tomcat (TOMCAT, 2008), que para o contexto desta dissertação pode ser definido como um serviço hospedeiro para extensões de servidores (*servlets*) em linguagem Java a serem publicadas através do uso de protocolo HTTP. A partir deste momento, é possível solicitar a execução de operações externas do serviço através do desenvolvimento de uma aplicação cliente Java que solicite a execução dos métodos pertinentes.

A Figura 12 apresenta a arquitetura do Apache Muse de forma simplificada. Pode-se notar também a presença de um roteador de recursos, que direciona as solicitações entre aplicação cliente e recursos, como um intermediário, e do gerenciador de recursos.

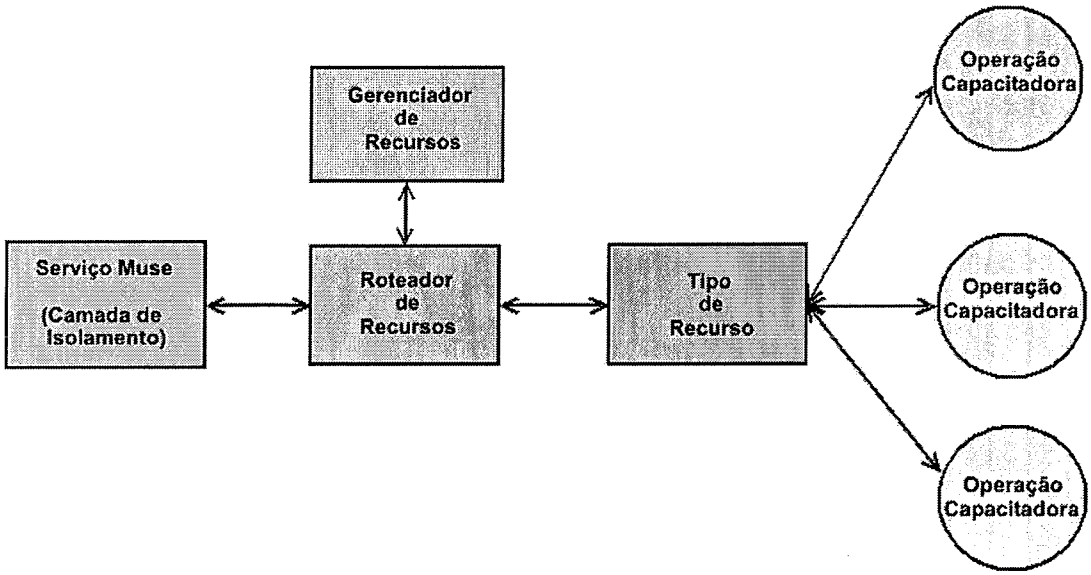

Figura 12 – Elementos da Arquitetura do Apache Muse

Fonte: (MUSE, 2008).

Todo serviço desenvolvido no Apache Muse pode ser concebido de forma a tornar-se de natureza WSRF ou não. É importante ressaltar que isto não significa que o comportamento esperado para cada operação definida pelo WSRF será realizado automaticamente. Existe apenas a previsão de uso das mesmas. Este é um comportamento interessante, pois propicia liberdade ao desenvolvedor, mas que também agrega um grau de dificuldade, que envolve o projeto do documento de propriedades do serviço, do esquema do mesmo, do seu documento WSDL e das operações WSRF que serão realizadas.

Estas ações ainda carecem de mecanismos facilitadores para o desenvolvimento de serviços dessa natureza, apesar de existirem opções como aplicativos agregadores de funcionalidades (*plugins*) para a plataforma Eclipse (ECLIPSE, 2008), que podem auxiliar em tarefas básicas do desenvolvimento, como na escrita de um documento WSDL com uso de uma interface gráfica, por exemplo. O Eclipse TPTP (*Test & Performance Tools Plataform*) (ECLIPSE TPTP, 2008) é um destes aplicativos e foi utilizado para o desenvolvimento do GParGRES. Uma vez selecionadas as operações de

um recurso, é possível implementá-las através da inclusão de bibliotecas do Muse que realizam a interface entre o WSRF e a aplicação em desenvolvimento.

O Apache Muse foi selecionado como base para o desenvolvimento do GParGRES dentre as opções disponíveis na comunidade por diversos fatores: código aberto; implementação das especificações de cada padrão da WSRF; independência de mediador de grade, pois serviços Muse podem ser expostos através do Tomcat/Axis2 apenas, sem necessidade de configurações específicas (requer apenas Java 1.5 e Tomcat 5.5, e as bibliotecas do Muse); utilização da linguagem Java, como o ParGRES e o GParGRES, de programação utilizada igual à do ParGRES e GParGRES, inclusive para o desenvolvimento de aplicações cliente.

O uso de operações externas (*capabilities*) permite que seja possível definir cada uma das operações do GParGRES, a ser exposta à aplicação cliente, como o pedido para execução de uma consulta em linguagem SQL, por exemplo. Demais operações podem ou não ser expostas, a critério do desenvolvedor. Estas operações podem ser agrupadas em recursos tipicamente WSRF, quando pertinentes, ou em outros recursos não WSRF, caso desejado, sem que haja prejuízo ou problemas de integração entre os recursos de um serviço.

2.4 – Iniciativas para Bancos de Dados em Grades

Conforme pode ser visto em (ANNIS, ZHAO et al., 2002), aplicações de usuários finais por muitas vezes necessitam tratar grandes massas de dados, de forma similar ao que ocorre atualmente em armazéns de dados, e em aplicações de bioinformática e física, por exemplo.

Atualmente, existem soluções integradas a grades com a função de manipular e trafegar arquivos de grande porte, como o GridFTP (ANNIS, ZHAO et al., 2002; GLOBUS, 2007), MaxBCG (ANNIS, ZHAO et al., 2002), entre outros. Com relação à integração e suporte a bancos de dados em grades, existem iniciativas em andamento, como o conjunto OGSA-DAI (ANJOMSHOAA, ANTONIOLETTI et al., 2005), OGSA-DQP (ALPDEMIR, MUKHERJEE et al., 2003), Spitfire (BELL, BOSIO et al., 2002), Oracle 11g (ORACLE, 2007; SHIMP e MIRANDA, 2005), entre outros.

Conforme pode ser observado em (WATSON, 2003), existem desafios para a integração de bancos de dados em ambientes de grade. Dentre os desafios citados, destacam-se os seguintes:

1. Como prover suporte a diferentes SGBD dentro de um mesmo ambiente (a grade), de forma que seja possível submeter consultas aos mesmos, de forma transparente ou não, a partir de aplicativos de usuários da comunidade da grade, além de garantir a composição de resultados provenientes de diferentes bases de dados e SGBD distintos, com suporte a paralelismo de consultas?

2. Como prover suporte a diferentes modelos de dados? Cada SGBD, seja relacional, orientado a objetos, XML, entre outros, tem sua própria arquitetura, definição de interfaces e modelos de dados. Em um ambiente de grade, pressupõe-se o conceito de heterogeneidade e, portanto, podemos observar distintos SGBD no mesmo. Sendo assim, é necessário endereçar cada um dos mesmos, tornando-os aplicativos suportados pelo ambiente de grade e assim disponíveis para seus usuários finais.

Uma das alternativas para prover tal integração e suporte é através de mediadores genéricos, independentes de SGBD e modelos de dados, que sejam capazes de interpretar requisições dos usuários, coordenarem a distribuição de tarefas para cada nó da grade e garantir a composição do resultado final e respectivo retorno ao usuário.

Existem diversos outros fatores característicos de SGBD, entre os quais podemos citar: questões de segurança, escalabilidade, natureza das aplicações dos usuários (OLTP x OLAP) e consequente otimização. Todos estes fatores, entre outros, precisam ser abordados dentro do contexto de grades computacionais.

O uso de bancos de dados em grades pode agregar valor aos aplicativos dos usuários do mesmo, além de beneficiar-se do desempenho computacional fornecido pela grade. Algumas das possibilidades para o uso de bancos de dados em tais ambientes listadas por esta dissertação são listadas nas subseções a seguir, e envolvem acesso a metadados, proveniência de dados, manipulação de bases de conhecimento e repositórios de projetos corporativos, entre outros.

2.4.1 – OGSA-DAI

OGSA-DAI (ANJOMSHOAA, ANTONIOLETTI et al., 2005; PATON, ATKINSON et al., 2002) é um mediador com base na arquitetura OGSA, elaborado para fornecer acesso a bases de dados relacionais e XML em grades. Entende-se que para o GParGRES a OGSA-DAI teria uma função complementar, visto que apenas expõe bases de dados na grade e não fornece o processamento das consultas da natureza e na forma desejada, com paralelismo e distribuição de carga entre os nós. A OGSA-DAI fornece um método padrão para submeter uma consulta a um denominado recurso de dados para grade e obter o resultado correspondente, de forma similar ao processo realizado por uma aplicação ou usuário em um ambiente centralizado e sequencial. Por outro lado, a OGSA-DAI pode ser utilizada como referência no momento de projetar as operações do GParGRES.

Conforme pode ser visto em (ANJOMSHOAA, ANTONIOLETTI et al., 2005; DAIS, 2007; PATON, ATKINSON et al., 2002), os objetivos do OGSA-DAI compreendem:

1. Prover exposição controlada dos recursos de dados a grade, suportando bases de dados heterogêneas, de forma a garantir a prestação de serviços básicos de recuperação.
2. Fornecer serviços básicos que permitam a construção de serviços de integração de dados mais alto nível, que disponibilizem, por exemplo, federação de dados, ou suportem adoção de demais funcionalidades projetadas ou solicitadas pelo usuário final.
3. Padronização das interfaces de acesso a dados.
4. Fornecer uma implementação de referência para a especificação de serviços de acesso e integração de dados DAIS, que tem como objetivo fornecer um padrão para serviços de grade que garantam acesso persistente à bases de dados através de serviços *web* (DAIS, 2007).

O OGSA-DAI provê três serviços para atender a tais requisitos. Estes serviços são:

1. GDSF – *Grid Data Service Factory* – fábrica responsável por instanciar os GDS para os usuários da grade e expor os recursos de dados disponíveis e seus respectivos metadados ao cliente.
2. GDS – *Grid Data Service* – responsável por interagir com os recursos de dados disponíveis na grade, o GDS atende às solicitações dos usuários, as despacha para os SGBD e monta os resultados provenientes das consultas executadas, exibindo-as para o usuário final.
3. DAISGR – *DAI Service Group Registry* – permite aos GSDF registrarem-se e anunciarem suas capacidades e metadados que auxiliem na descoberta de serviços.

A Figura 13 apresenta uma possibilidade de fluxo de execução de um serviço para grade.

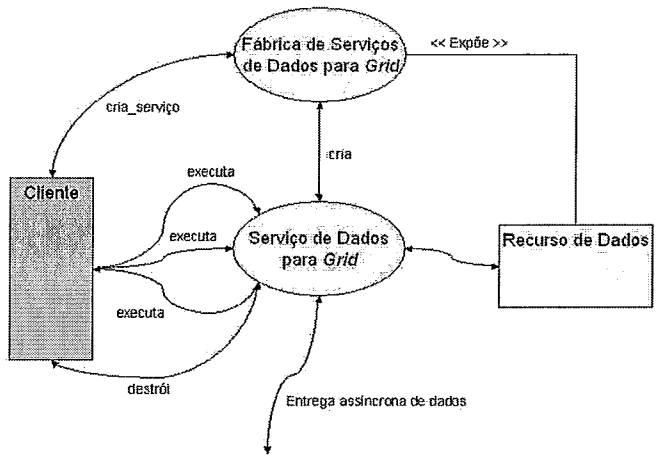

Figura 13 - Exemplo de Fluxo de Execução

Fonte: (ANJOMSHOAA, ANTONIOLETTI et al., 2005)

Cada GDSF expõe somente um único recurso de dados por vez, além do mesmo ser estaticamente configurado por um arquivo designado para tal função.

Além disso, os GDS não são persistentes e o GDSF não tem a capacidade de processar as consultas diretamente. Este instancia GDS que podem executar as

operações determinadas através do arquivo de configuração citado previamente, sendo que as operações são diferentes para cada paradigma de banco de dados (relacional ou XML).

2.4.2 – OGSA-DQP

Enquanto a OGSA-DAI (ANJOMSHOAA, ANTONIOLETTI et al., 2005; PATON, ATKINSON et al., 2002) expõe recursos de dados para capacitar a execução de consultas em ambiente de grade, de forma similar ao processo tradicionalmente centralizado e sequencial, uma alternativa para o processamento distribuído de consultas é apresentada pela OGSA-DQP (ALPDEMIR, MUKHERJEE et al., 2003), baseada na OGSA-DAI. Mesmo assim, a OGSA-DQP não provê automaticamente o paralelismo intra-consulta no nível de operador.

São três os componentes principais da OGSA-DQP:

1. GDQSF – *Grid Distributed Service Factory* – funciona de forma similar ao GDSF, da OGSA-DAI, sendo a fábrica responsável por registrar-se no GDSR (DAISGR) e atender às solicitações de clientes para criação de novas instâncias de GDQS.
2. GDQS – *Grid Distributed Query Service* – estende a interface do GDS para obter metadados e informações de recursos computacionais para atuar sobre os planos de execução de consultas distribuídas. Esta implementação foi realizada com base no projeto POLAR (POLAR, 2007; SMITH, WATSON et al., 2000), no que diz respeito à compilação e otimização de consultas.
3. GQES – *Grid* Query *Evaluation Service* – também estende o GDS e é responsável por executar uma partição do plano de execução designado pelo GDQS, ou seja, atua para viabilizar a execução das consultas, repassando-as aos GDS propriamente ditos, que por sua vez, executam a sub-consulta sob o SGBD em questão.

A Figura 14 demonstra a arquitetura do OGSA-DQP e ilustra o passo-a-passo para execução de consultas com paralelismo na grade.

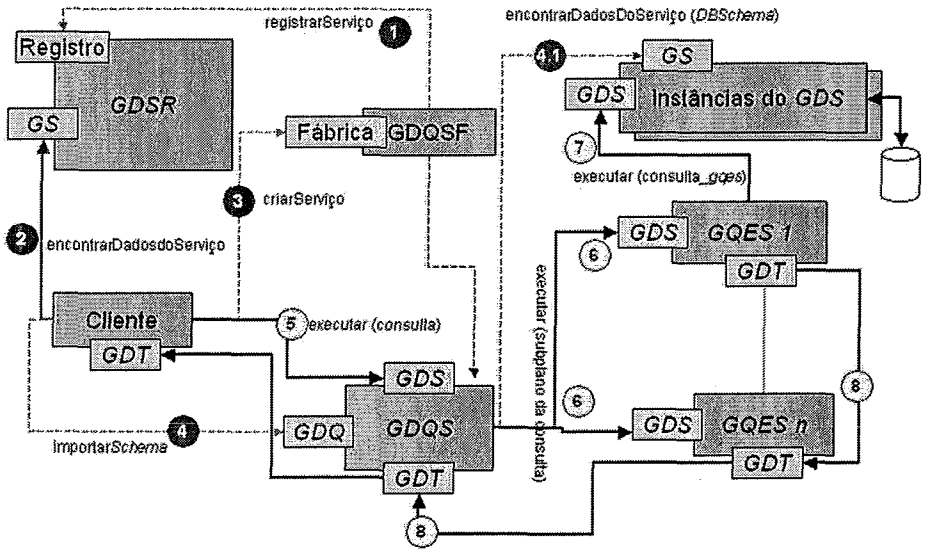

Figura 14 – Arquitetura e Fluxo de Execução do OGSA-DQP

Fonte: (ALPDEMIR, MUKHERJEE et al., 2003)

O GParGRES fornece o mesmo pois suporta o particionamento de dados, o que é essencial para o processamento de consultas de apoio à decisão. Com o paralelismo em questão, o mesmo operador (por exemplo, um operador de junção) é executado em paralelo pelo GParGRES através de diferentes instâncias do ParGRES que processam subconjuntos de dados. Sendo assim, o GParGRES oferece melhor alternativa para consultas de apoio à decisão.

2.4.3 – Spitfire

Outra iniciativa no sentido de criação de mediadores para integração de bancos de dados a ambientes de grades corresponde ao projeto Spitfire (BELL, BOSIO et al., 2002). Este é um mediador, na forma de um serviço *web*, que recebe requisições baseadas em XML via HTTP e as repassa aos SGBD (relacionais) com o uso de controladores de acesso JDBC. A arquitetura da solução Spitfire pode ser visualizada na Figura 15.

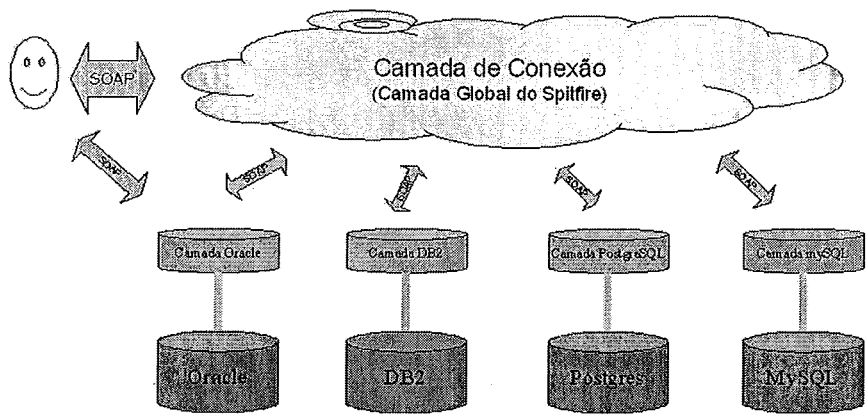

Figura 15 - Projeto Spitfire

Fonte: (BELL, BOSIO et al., 2002)

Este projeto fez parte do European Data *Grid*, concluído em Março de 2004. A última atualização do Spitfire data de Dezembro de 2003. A relevância deste projeto para esta dissertação é de fator histórico e relaciona-se para com o projeto em questão ter direcionado seus esforços para que a fosse concebido como um serviço *web*. Além disso, teve a participação de colaboradores do projeto OGSA-DAI, o que reforça a motivação de criar o GParGRES como um serviço para grade, que estende as características de um serviço *web* com as funcionalidades já listadas anteriormente.

2.4.4 – Oracle 11g

A partir de outro conceito de grades, em relação ao utilizado na comunidade científica, a Oracle (ORACLE, 2007) busca através do produto Oracle 11g integrar seu produto comercial em tal infra-estrutura.

Conforme pode ser visto em (SHIMP e MIRANDA, 2005), para a Oracle, grade corresponde a "agrupar recursos de tecnologia da informação em um grupo de serviços compartilhados para atingir às necessidades computacionais de uma organização".

A solução de grade adotada pela Oracle tem por base os conceitos de *virtualização* e *provisionamento*, que em certos momentos remetem ao conceito de Organizações Virtuais, apresentados anteriormente por (FOSTER, KESSELMAN et al., 2001) e (CARPENTER e JANSON, 2004). Por virtualização, entende-se a quebra de elos estritos entre quem provê e quem consome recursos, que estão agrupados na grade e disponibilizados aos usuários do mesmo. Já o provisionamento compreende a capacidade inerente da grade em determinar como atender às solicitações do usuário, através da disponibilização de recursos computacionais.

Para o desenvolvimento de aplicações que se beneficiem do Oracle 11g, a Oracle recomenda a implementação em serviços *web* com base em XML. A criação da grade com o suporte do Oracle 11g envolve três passos, a saber:

1. Consolidação de recursos de hardware, aplicações e informação da organização.
2. Padronização de recursos de servidores, armazenamento e sistemas operacionais; de aplicações, através de serviços *web*; de fontes de informações e metadados.
3. Automação de tarefas do dia-a-dia.

Neste momento pode-se notar uma diferença no conceito de grade referenciado nesta dissertação e do considerado pela Oracle. Por exemplo, para a Oracle existe a preocupação em garantir homogeneidade entre os recursos que compõem a grade, o que nos aproxima da idéia de agrupamentos.

2.4.5 – Armazéns de Dados

Existem trabalhos relacionados que propõem novos modelos de dados para armazéns de dados em grades, como (WERHLE, MIQUEL et al., 2005). Através da fragmentação de dados (BELLATRECHE, KARLAPALEM et al., 1999), fragmentos físicos do armazém de dados são distribuídos entre os nós da grade. Em seguida, serviços para grade são construídos para identificar e indexar tais fragmentos. Alguns serviços para grade também são propostos para a criação de planos de consulta distribuídos. Uma das vantagens do GParGRES é fornecer o paralelismo inter e intra-consulta sem requerer qualquer fragmentação física da base de dados. Adicionalmente, o GParGREs trabalha com SGBD relacionais tradicionais, enquanto a abordagem apresentada em (WERHLE, MIQUEL et al., 2005) não deixa claro se os dados são armazenados em bancos de dados relacionais ou arquivos.

Em (LAWRENCE, DEHNE et al., 2007; LAWRENCE e RAU-CHAPLIN, 2006), é proposto um algoritmo para processamento de consultas de apoio à decisão (OLAP) em grades, com duas fases, que compõe a solução denominada pelos autores como "grade habilitada para OLAP" (*OLAP Enabled Grid*). A primeira fase conta com a recuperação de dados que estejam presentes na memória *cache* de cada servidor OLAP local disponível; fragmentos de dados indisponíveis localmente são identificados, para que então sejam elaboradas sub-consultas, a serem executadas por nós remotos, que venham a ser detentores de tais fragmentos. Um único esquema de banco de dados é compartilhado entre todos os servidores disponíveis e os dados são particionados horizontalmente.

Os autores apresentam resultados apontados como positivos, com aceleração de 50% a 60% na execução das consultas, mas faltam detalhes como: o projeto de fragmentação; a escala dos recursos físicos (*hardware*) utilizados (quantidade de nós, características dos mesmos, etc.); qual *benchmark* utilizado; quais as consultas submetidas e suas características.

De igual importância, mesmo com citações referentes à criação de uma arquitetura de serviços com base na OGSA (FOSTER, KESSELMAN et al., 2001), os experimentos foram realizados somente através de um protótipo em linguagem de rotinas Python, com servidores OLAP e bases de dados simuladas. De qualquer forma,

faltam detalhes sobre como os experimentos foram realizados, que nos permitam analisar melhor o desempenho apontado pelos autores.

Finalmente, a abordagem proposta em (LAWRENCE, DEHNE et al., 2007; WERHLE, MIQUEL et al., 2005) utiliza índices espaciais construídos em Árvores-X. Esta estrutura de índices não é encontrada com facilidade em SGBD, visto que requer implementação com um propósito específico. O GParGRES não requer qualquer estrutura especial de índice. Ele requer apenas índices ordenados agrupados, o que pode ser encontrado com facilidade em diversos SGBD.

2.5 – Benchmark TPC-H

2.5.1 – Definição

A seguir apresentamos o *benchmark* utilizado para avaliação do GParGRES. Conforme pode ser visto em (TPCH, 2008), o *benchmark* TPC-H é projetado com foco em consultas de apoio à decisão, engloba um conjunto de 22 consultas em padrão SQL92 (SQL92, 2008), de natureza *ad-hoc*. Segundo (TPCH, 2008), "este *benchmark* ilustra sistemas de apoio à decisão que: operam sobre grandes volumes de dados, executam consultas com elevado grau de complexidade e oferecem respostas às questões críticas de negócio". Ainda, sobre as consultas que dele fazem parte, afirma que "oferecem respostas às questões do mundo real de negócios, simulam consultas *ad-hoc* (por exemplo, através de uma interface de apontar e clicar), são mais complexas do que transações OLTP, incluem uma extensa gama de operadores e restrições (*constraints*) de seleção, gera atividade intensa no SGBD durante a execução de experimentos, são executadas perante uma base de dados com população e dados de escalabilidade específicos e implementadas com restrições (*constraints*) próximas às utilizadas em um ambiente de produção".

2.5.2 – Base de Dados

O esquema da base de dados utilizada pelo TPC-H envolve a criação de oito tabelas, sendo estas: seis tabelas de dimensão (*customer*, *nation*, *part*, *partsupp*, *region* e *supplier*) e duas tabelas de fato (*lineitem* e *orders*). A Figura 16 ilustra a estrutura das tabelas em questão.

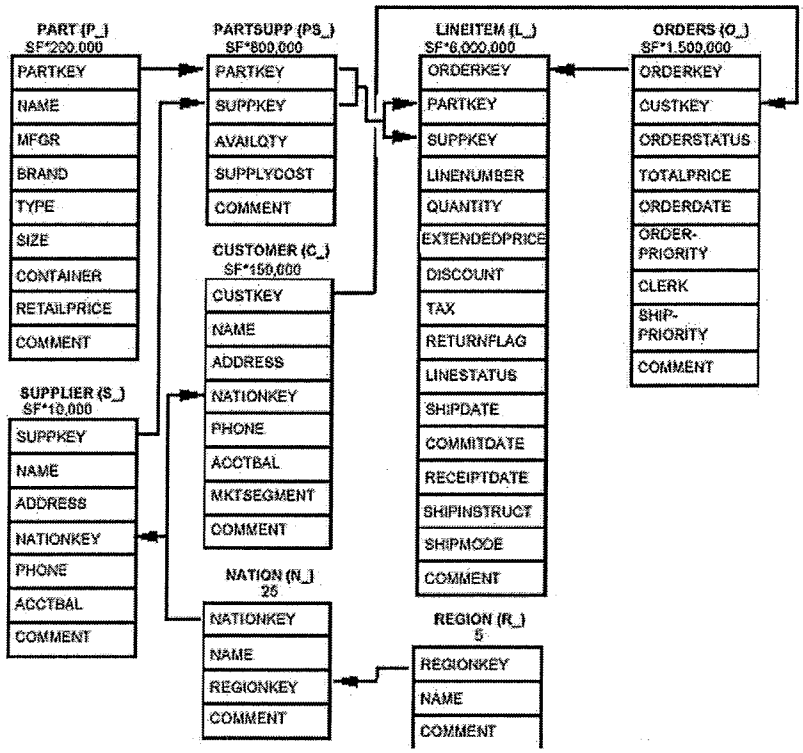

Figura 16 – Estrutura das tabelas do *benchmark* TPC-H

Fonte: (TPCH, 2008).

Nesta mesma figura pode-se observar a regra de formação da cardinalidade de cada uma das tabelas. Enquanto as tabelas *nation* e *region* possuem cardinalidade fixa (5 e 25, respectivamente), as demais são múltiplas do fator de escala, parâmetro utilizado pelo TPC-H para determinar a cardinalidade de cada relação por ele definida.

O próprio usuário do *benchmark* escolhe qual o fator de escala desejado para a criação das relações (SF - *scale factor*), através de um programa do próprio *benchmark* para a geração das mesmas, o DBGEN (TPCH, 2008).

Supondo o fator de escala igual a 1, recomendado pelo TPC-H para validação da criação das relações, temos a criação de tabelas da base de dados com as características

da Tabela 4. As duas tabelas de fato representam aproximadamente 80% do total de tuplas da base de dados.

Tabela 4 – Tamanho estimado das tabelas da base de dados

Tabela	Cardinalidade (linhas)	Tamanho de cada linha (em bytes)	Tamanho típico da tabela (em MB)
SUPPLIER	10.000	159	2
PART	200.000	155	30
PARTSUPP	800.000	144	110
CUSTOMER	150.000	179	26
ORDERS	1.500.000	104	149
LINEITEM[1]	6.001.215	112	641
NATION	25	128	< 1
REGION	5	124	< 1
Total	8.661.245	N/A	958

[1] – A regra para a cardinalidade desta tabela é simplificada para SF * 6.000.000, mas não é um múltiplo exato, uma vez que o número de tuplas de *lineitem* associadas à *orders* é escolhido de forma randômica com média de quatro. Para tal, o benchmark oferece o número de tuplas de forma pré-determinada para cada fator de escala autorizado.

Capítulo 3 – GParGRES

3.1 – Definição

GParGRES – ParGRES *Grid Service* – representa uma evolução do ParGRES, direcionada para ambientes de grades. É uma camada de aplicação em forma de mediador (*middleware*) para execução de consultas de apoio à decisão (OLAP), de forma paralela e distribuída em ambientes de grade. O GParGRES provê paralelismo inter - e intra-consulta, de forma não intrusiva aos SGBD e bases de dados relacionais, alocados ambos em cada uma das máquinas que compõem um nó da grade (no caso desta dissertação, a *Grid'5000* (GRID'5000, 2008)). Assumimos que cada nó da grade é composto por um agrupamento de PC.

Foi estudada a viabilidade de utilizar técnicas e padrões estabelecidos na comunidade científica para o desenvolvimento de serviços para grades, como a OGSA (FOSTER, KESSELMAN et al., 2002) e o arcabouço WSRF (WSRF, 2008), para a implementação do GParGRES. Dessa forma, facilita-se o uso do GParGRES em ambientes de grades, devido à possível portabilidade adotada pelos padrões estudados.

O GParGRES foi implementado sobre a plataforma *Grid'5000* (CAPPELO, DESPREZ et al., 2005; GRID'5000, 2008), como uma camada de aplicação caracterizada como um mediador, que fornece balanceamento de carga, através de mecanismo de fragmentação virtual simples (SVP) (AKAL, BÖHM et al., 2002), paralelismo inter - e intra-consulta, transparência e não-intervenção aos SGBD, aderência ao arcabouço WSRF (WSRF, 2008) e que trabalha em conjunto com instâncias do ParGRES no ambiente de grade.

Cada instância de SGBD e base de dados associada são orquestrados por instâncias da aplicação ParGRES através de técnicas não-intrusivas (caixa-preta). O GParGRES coordena instâncias do ParGRES. Nossa abordagem apresenta dois níveis de divisão de consultas: na grade, implementada pelo GParGRES e em seus nós, propiciada pelo ParGRES.

O GParGRES assume replicação total das bases de dados entre os diferentes nós da grade, conforme se observa no ParGRES, base para o desenvolvimento do GParGRES. Mesmo assim, a replicação parcial poderia ser aplicada através da alocação

de diferentes conjuntos de dados para cada nó da grade, enquanto ao mesmo tempo, todos adotam uma mesma definição de estrutura de base de dados. A replicação parcial das bases de dados envolve diversos aspectos que não fazem parte do escopo desta dissertação. Em tal cenário, o balanceamento de carga na grade representa um grande desafio, uma vez que cada sub-consulta realocada entre os nós pode requerer a migração de dados entre os mesmos, de forma a garantir seu processamento.

Dentre os padrões para desenvolvimento de serviços para grade citados, optamos pela adoção do arcabouço WSRF, em conjunto com a sua implementação realizada pelo projeto Apache Muse (MUSE, 2008). Esta escolha foi motivada pelos fatores descritos a seguir.

- O WSRF propicia a possibilidade de desenvolver um conjunto de serviços com independência de mediador de grade e de capacitar cada um dos serviços com diversas características relativas a serviços para grades, como: atribuir estado a um serviço, gerenciar o ciclo de vida do serviço, de forma que o desenvolvedor pode optar pela destruição do mesmo pelos critérios que julgar necessário, pelo possível agrupamento dos serviços e optar pela gestão de falhas dos mesmos.
- Ao mesmo tempo, o Apache Muse representa uma implementação do arcabouço WSRF, de código aberto, e especifica cada uma das operações definidas pelo WSRF. Além disso, as facilidades oferecidas pelo ambiente de desenvolvimento associado, com suporte do Apache Tomcat 5.5/Axis2, linguagem Java e as próprias bibliotecas do Muse é cativante e próxima do ParGRES, que utiliza a referida linguagem Java como base para seu desenvolvimento.
- A arquitetura do Muse permite que cada recurso projetado seja inserido em uma operação capacitadora (*capability*), que é o nome dado a unidades de programação (módulos) que têm a possibilidade de expor operações ao mundo exterior (aplicação cliente) ou não, a critério do desenvolvedor. Estas capabilities também possuem operações internas, não visíveis ao mundo externo. Estas operações externas podem também ser desenvolvidas com a concepção de um recurso compatível com o arcabouço WSRF ou não, a critério do desenvolvedor, sem que ocorram problemas de integração entre os serviços.

Dentre as soluções apresentadas nesta dissertação, não foi possível observar o uso conjunto de todas estas funcionalidades listadas. Com exceção das ferramentas Oracle 11g (ORACLE, 2007), que não especifica detalhes de sua implementação, fato provavelmente relacionado a natureza de código fechado da mesma e do modelo apresentado por (WERHLE, MIQUEL et al., 2005), que também não lista como a solução foi desenvolvida internamente, todas as demais foram concebidas como serviços *web*, mas não trazem as características de um serviço para grade, como dinamicidade, gerência de estado e ciclo de vida, oferecidas pelo arcabouço WSRF (WSRF, 2008) e sua respectiva implementação pelo projeto Apache Muse (MUSE, 2008).

O GParGRES, portanto, representa uma abordagem diferenciada em relação as demais soluções aqui apresentadas. Assim como o GParGRES concretiza seus componentes como serviços para grade, também agrega processos como o paralelismo inter- e intra-consulta e a composição de resultados, que serão detalhados neste capítulo, na seção 3.3 –. Finalmente, toda a concepção do GParGRES foi realizada de forma independente de mediador de grade (por exemplo, o Globus Toolkit (GLOBUS, 2007)).

3.2 – Arquitetura

O GParGRES foi idealizado através de um conjunto de serviços compatíveis com o arcabouço WSRF. A arquitetura proposta para o GParGRES foi publicada em (KOTOWSKI, LIMA et al., 2007; KOTOWSKI, LIMA et al., 2008) e contempla um conjunto de quatro serviços, ilustrados na Figura 17.

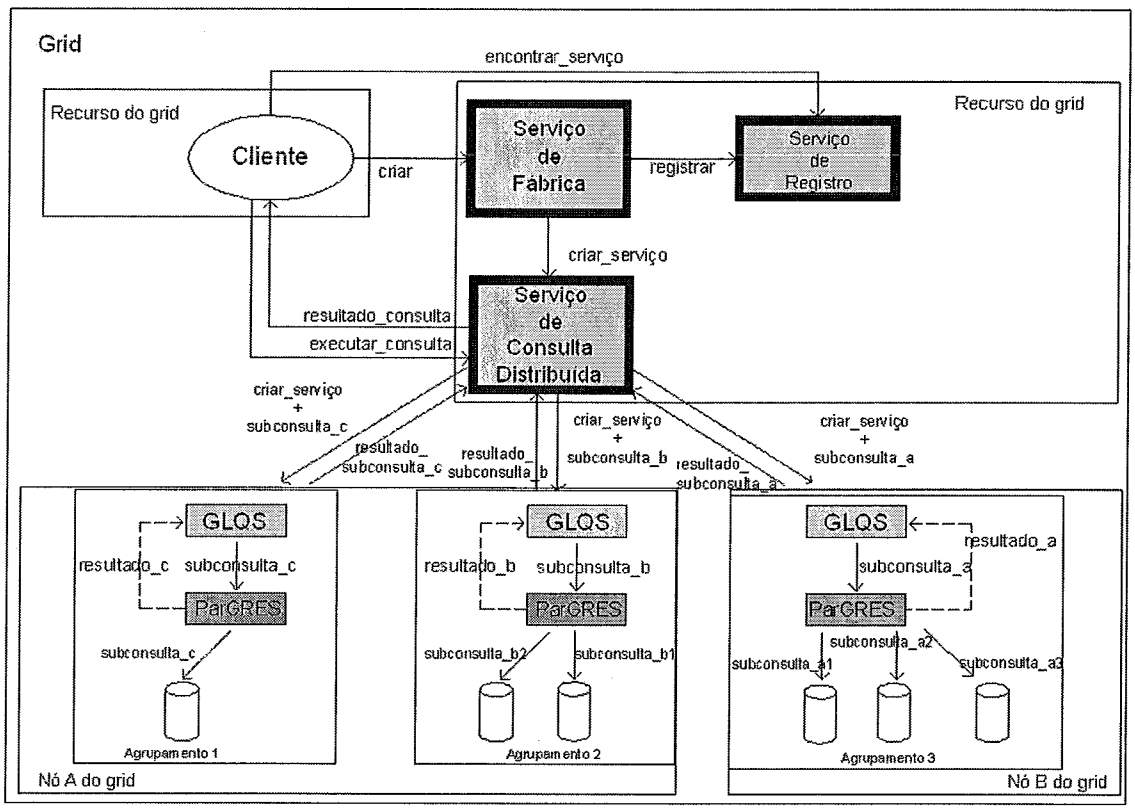

Figura 17 – Arquitetura do GParGRES

Fonte: (KOTOWSKI, LIMA et al., 2007; KOTOWSKI, LIMA et al., 2008).

Na figura em questão, podemos notar a alocação de recursos na grade para cada um dos componentes envolvidos na execução de uma consulta submetida ao GParGRES. Como exemplo, temos a participação de três agrupamentos distintos, sendo que o "Agrupamento 1" e o "Agrupamento 2" pertencem ao "Nó A", enquanto o "Agrupamento 3" pertence ao "Nó B".

Também temos dois recursos da grade separados dos demais. Um dos mesmos representa o local onde o cliente é instanciado. Por cliente, queremos dizer uma aplicação que solicite a execução de uma consulta ao GParGRES. O outro recurso

concentra três dos serviços da arquitetura do GParGRES. A seguir, detalhamos cada um dos mesmos.

Serviço de Fábrica - FS (*Factory Service*) – responsável por criar novas instâncias do serviço DQS. Quando uma aplicação cliente deseja submeter consultas a grade através do GParGRES, ela solicita a criação de um DQS através de um serviço FS, que é instanciado dentro do concentrador de operações (*port type*) do GParGRES, no mesmo recurso da grade onde a aplicação cliente foi disparada. Cada DQS é associado a uma solicitação e seu ciclo de vida compreende o período necessário para completar as solicitações da aplicação cliente que requisitou sua criação através do FS.

O serviço FS também possui um identificador único, composto do endereço URI do mesmo, associado a um identificador de recursos (*resourceid*) criado pela própria aplicação Muse. O FS possui função similar a fábrica (*Grid Data Service Factory* (GDSF)) da arquitetura OGSA-DAI (ANJOMSHOAA, ANTONIOLETTI et al., 2005), que corresponde principalmente à criação de instâncias do serviço principal da aplicação GParGRES, no caso o DQS.

Serviço de Consulta Distribuída – DQS (*Distributed Query Service*) – este serviço, depois de criado pelo FS em um dos recursos que compõem a grade, tem a responsabilidade de atender à requisição da aplicação cliente, o que representa solicitar a execução da consulta de apoio à decisão aos recursos da grade e garantir que o resultado da mesma seja entregue a quem o solicitou. Atualmente, o DQS é criado no mesmo recurso em que o FS foi criado e a parte dos GLQS utilizados.

Enquanto adotar o mesmo recurso da grade para criação dos serviços FS e DQS não caracteriza problema, visto que suas operações, apesar de essenciais, consomem recursos de forma moderada, as operações dos GLQS representam elevado consumo de memória, pela interação direta para com as instâncias de ParGRES e consequentemente, bases de dados.

Dessa forma, alocar o DQS e os GLQS em recursos distintos pode trazer como benefício a melhor utilização da área de memória dos mesmos, para cada ação realizada por cada um dos serviços do GParGRES, através da não interferência entre DQS e GLQS, o que caracteriza um melhor isolamento dos processos, se comparada a uma arquitetura na qual todos os serviços estariam atribuídos a um mesmo recurso.

Este processo envolve diversos passos e podemos dizer que o DQS é o serviço responsável por coordenar a execução da consulta, o que significa recebê-la, operar o mecanismo de processamento e paralelismo adotado pelo GParGRES, repassar as devidas sub-consultas geradas para execução nos recursos apropriados da grade, atuar na composição dos resultados parciais de cada sub-consulta de forma a produzir o resultado final esperado, e entregar tal resultado à aplicação que solicitou a execução da consulta de apoio à decisão.

Para realizar todas estas operações, o DQS recebe como parâmetro um arquivo de configuração em formato que segue a linguagem XML. A seguir apresentamos um exemplo de arquivo de configuração para o GParGRES. Este arquivo é dividido em seções e contém todos os parâmetros necessários para a execução da consulta.

Arquivo de Configuração GParGRES:

```
<gpargres>
   <query_info>
        <query the_query="/local/gpargres/query.sql"
            glqs_list="pastel-37.toulouse.grid5000.fr;
            paramount-32.rennes.grid5000.fr" />
   </query_info>
   <general_settings>
        <settings  scale_factor="1"  is_intra_query="true"  />
    </general_settings>
   <result_composition>
     <result the_create_table="/local/gpargres/create_table.sql"
            the_select_command="/local/gpargres/select.sql" />
   </result_composition>
</gpargres>
```

A seção inicial, (*query_info*), concentra informações referentes à consulta ser submetida a grade. Nela podemos observar os seguintes parâmetros:

- Arquivo com a consulta (*the_query*) – contém um ponteiro para o arquivo em linguagem SQL que possui a consulta a ser submetida a grade. Deve ser alterado pelo usuário conforme mude a consulta a ser executada.

- Lista de GLQS (*glqs_list*) – assim como no ParGRES devemos informar previamente em que recursos do agrupamento os mediadores serão criados (componentes CQP), no GParGRES é necessário informar em que recursos de cada nó da grade os GLQS serão criados.

A seção seguinte (*general_settings*) está relacionada a configurações gerais requeridas pelo GParGRES e lida com o *benchmark* TPC-H e o paralelismo da consulta. Os parâmetros utilizados são:

- Fator de escala do *benchmark* TPC-H (*scale_factor*) – o fator de escala, conforme apresentado na seção 2.5 –, está diretamente relacionado à cardinalidade da base de dados utilizada e terá papel fundamental no processamento da consulta, para a correta criação dos intervalos dos atributos de fragmentação (*l_orderkey, o_orderkey*).
- Informação se a consulta é ou não candidata a paralelismo intra-consulta (*is_intra_query*) – este parâmetro, com tipo de dados booleano, informa ao GParGRES se a consulta a ser executada deve ou não ser alvo da técnica de fragmentação virtual simples adotada. Caso a consulta não seja candidata ao paralelismo intra-consulta, ela será repassada diretamente ao primeiro GLQS criado para execução. Este comportamento é similar ao do ParGRES, no qual o mediador (CQP) solicita ao primeiro NQP disponível que execute a consulta caso a mesma seja candidata a paralelismo inter-consulta.

Finalmente, temos uma seção dedicada a parâmetros relacionados à composição de resultados (*result_composition*). O processo de composição será descrito posteriormente, de forma detalhada na seção 3.3.4 –. Temos dois parâmetros que representam:

- Comando para criação de tabela temporária (*the_create_table*) – este parâmetro aponta para um arquivo que contém o comando em linguagem SQL para que o DQS solicite a criação de uma tabela temporária em memória que será utilizada para guardar os resultados

provenientes da execução de cada sub-consulta pelos GLQS associados. Este comando é próprio para cada consulta a ser submetida a grade e, portanto, o arquivo deve ser atualizado pelo próprio usuário.

- Comando para recuperação de dados da tabela temporária (*the_select_command*) – também relacionado à composição de resultados, este parâmetro informa ao DQS o arquivo que contém o comando em linguagem SQL para que os resultados armazenados na tabela temporária em memória sejam recuperados corretamente. Assim como o parâmetro anterior, está relacionado a cada consulta em particular e assim sendo, o arquivo deve ser alterado pelo usuário conforme necessário.

Os parâmetros associados às seções de composição de resultados e de configurações gerais foram utilizados, pois o GParGRES não conta com um analisador sintático (*parser*) como o ParGRES. A adoção de um analisador sintático, fato de característica complexa, não fazia parte do escopo desta dissertação, mas poderia ser adotada no ambiente de grade através, por exemplo, da abordagem de análise sintática proposta em (MATTOSO, ZIMBRÃO et al., 2005c; MATTOSO, ZIMBRÃO et al., 2005b; MATTOSO, ZIMBRÃO et al., 2005a). Sua ausência não inviabiliza ou prejudica a execução das consultas de apoio à decisão através do GParGRES.

Dessa forma, realizamos a passagem de tais parâmetros diretamente ao GParGRES. Consideramos que este fato não influencia negativa ou positivamente na execução da consulta, pois dentre os passos associados a tais parâmetros, não pudemos observar operações com indícios de consumo de memória, processamento ou entrada e saída representativos.

Serviço de Execução Local da Consulta - GLQS (*Grid Local Query Service*) – componente responsável por receber a sub-consulta a ser executada pela instância de ParGRES a ele associada. Também recebe como parâmetros os intervalos de valores associados a cada um dos atributos de fragmentação utilizados pela sub-consulta (*l_orderkey*, para casos onde a tabela *lineitem* seja referenciada, *o_orderkey* quando a tabela *orders* seja utilizada, ou ambos em casos onde as duas tabelas façam parte da sub-consulta) já definidos. O GLQS atua como um intermediário entre o DQS e o ParGRES, e também possui participação na composição de resultados, a ser

apresentada ainda nesta seção. Este serviço foi concebido tendo como base o conceito de um serviços de dados para grade (*Grid Data Service* (GDS)), da arquitetura OGSA-DAI (ANJOMSHOAA, ANTONIOLETTI et al., 2005), que representa um serviço capaz de interagir com um recurso de dados, como um SGBD, por exemplo.

Para o GParGRES, cada GLQS representa a interação com instâncias de ParGRES em cada um dos agrupamentos que compõem cada um dos nós da grade. Dessa forma, cada GLQS é instanciado no mesmo recurso que contém uma instância de mediador do ParGRES (CQP), componente com a função de receber as requisições de execução de consultas.

Serviço de Registro - RS (*Registry Service*) – concentra informações sobre os demais serviços do GParGRES, tais como o estado dos mesmos e o endereço URI de cada um dos mesmos. Este serviço é criado no mesmo recurso da grade onde o FS e o DQS foram criados. O RS foi idealizado para ser concebido como um serviço compatível com o arcabouço WSRF, e de comportamento similar ao *Index Service* descrito no MDS4 (WS MDS – *Web Service – Monitoring and Discovery System*) (MDS4, 2008).

Até o momento, o RS não foi concretizado como um serviço para grade com base no MDS4, pois se assim fosse realizado não teríamos a generalidade e independência de mediador de grade, uma vez que até o momento, o MDS4 está incluso no Globus Toolkit 4 (GLOBUS, 2007).

Portanto, as funções do RS foram realizadas como serviços para grade do Apache Muse, mas ainda não dentro dos conceitos adotados pelo MDS4, pelo motivo citado anteriormente.

Uma outra função interessante para o RS está na possibilidade de coletar metadados de recursos computacionais da grade para auxiliar em um processo que ofereça balanceamento de carga. Estes metadados, no caso do *Grid'5000*, podem ser obtidos através da ferramenta de monitoração de recursos computacionais denominada Ganglia (GANGLIA, 2007; MASSIE, CHUN et al., 2004; SACERDOTI, KATZ et al., 2003). Esta ferramenta permite recuperar informações sobre consumo de memória, processador, rede e sobre entrada e saída de máquinas que façam parte dos nós da grade, a partir de uma base de dados centralizada e própria do Ganglia, alocada em um nó pré-determinado pelos gestores da grade. Até o momento, estas funcionalidades estão

previstas, mas ainda não foram concretizadas. Espera-se que possam ser agregadas como contribuições futuras ao GParGRES.

3.3 – Processos

Nesta seção destacamos os quatro processos principais realizados pela aplicação GParGRES, que caracterizam o funcionamento da mesma, desde o recebimento de um pedido de execução de consulta de apoio à decisão por meio de uma aplicação cliente até o envio do resultado final à mesma.

3.3.1 – Inicialização dos Serviços

O primeiro passo envolvido na utilização do GParGRES no ambiente de grade envolve a inicialização dos recursos e respectivas ações que o mesmo pode oferecer aos clientes finais, na forma de operações externas do Apache Muse. Estes recursos representam os serviços da arquitetura do GParGRES. Para garantir a correta inicialização dos recursos em questão é necessário que exista um documento de propriedades do GParGRES (*Resource Properties Document*), escrito em linguagem WSDL, que funciona como descritor de cada serviço prestado pela aplicação GParGRES. O editor do Eclipse TPTP (ECLIPSE TPTP, 2008) permite a geração de uma visão para facilitar o entendimento do documento WSDL da aplicação, em forma de um diagrama, apresentado na Figura 18.

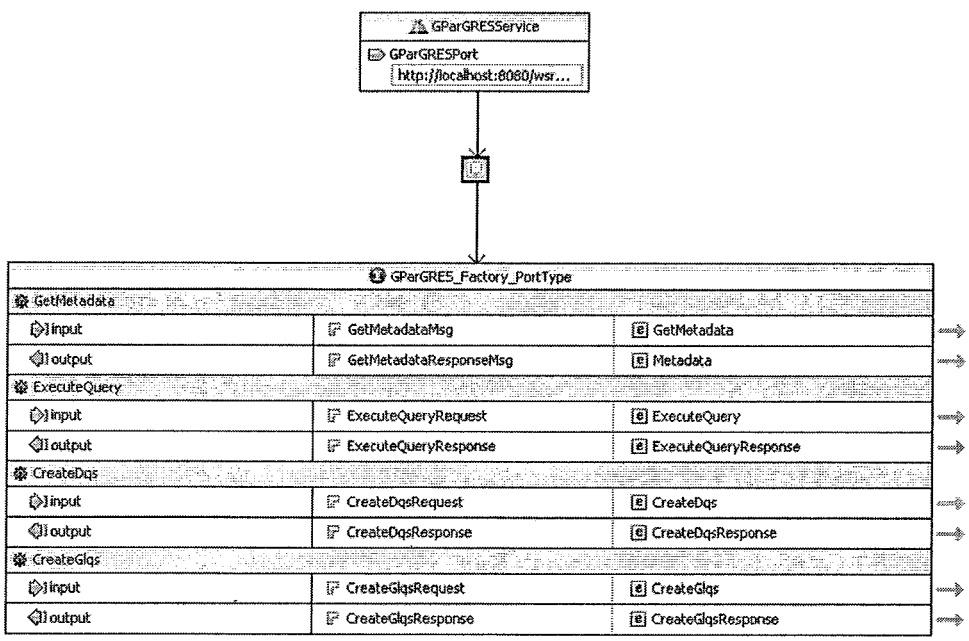

Figura 18 – Diagrama em WSDL do GParGRES

Na Figura 18, temos uma porta definida para expor o GParGRES através do uso do protocolo http (porta 8080) e o denominado tipo de porta (*port type*), um concentrador de operações fornecidas pelo GParGRES em alto nível. Pode-se observar a criação dos componentes DQS (*CreateDqs*) e GLQS (*CreateGlqs*) e da chamada para execução de sub-consultas (*ExecuteQuery*), que é executada por cada GLQS para cada instância de ParGRES a ele associado.

A chamada para execução de sub-consultas não é um serviço propriamente dito do GParGRES, mas também deve ser definida no documento em questão por requisito do Apache Muse, assim como a obtenção de metadados de cada um dos serviços também está presente neste documento, na forma do serviço *GetMetadata*.

O componente FS é criado automaticamente a partir do momento em que o servidor hospedeiro Tomcat é iniciado, e está definido em documento WSDL à parte, para facilitar o entendimento de que ele é o responsável por iniciar os demais componentes. A Figura 19 apresenta o diagrama WSDL do serviço FS. Ele contém uma chamada para criação do serviço DQS (*CreateDqs*), que solicita ao GParGRES a criação do DQS por operação de nome similar.

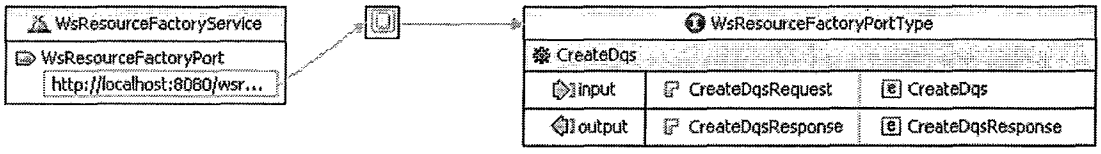

Figura 19 – Diagrama WSDL do serviço FS

Estes diagramas facilitam o entendimento do código em WSDL para leitura final, uma vez que a extensão de um arquivo com poucas operações já ganha destaque pela quantidade de informações associadas ao documento em questão. De qualquer forma, é possível editar o documento WSDL através de linha de comando, em um padrão similar à linguagem XML. O trecho a seguir representa o documento WSDL do serviço FS.

Documento WSDL do serviço FS:

```
<?xml version="1.0" encoding="UTF-8"?>

<wsdl:definitions
       targetNamespace="http://ws.apache.org/muse/test/wsrf"
       xmlns:tns="http://ws.apache.org/muse/test/wsrf"
       xmlns="http://schemas.xmlsoap.org/wsdl/"
       xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
       xmlns:wsdl-soap="http://schemas.xmlsoap.org/wsdl/soap/"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       name="WsResourceFactory">

       <wsdl:types>

       <xsd:schema
              elementFormDefault="qualified"
              targetNamespace="http://ws.apache.org/muse/test/wsrf/dqs">
              <xsd:element name="CreateDqs">
                     <xsd:complexType>
                     <xsd:sequence>
                     <xsd:element  name="gpargres_settings"  type="xsd:string"/>
                     </xsd:sequence>
                     </xsd:complexType>
              </xsd:element>
              <xsd:element name="CreateDqsResponse" type="xsd:string" />
       </xsd:schema>

       </wsdl:types>

       <wsdl:message name="CreateDqsRequest">
              <wsdl:part name="CreateDqsRequest" element="CreateDqs" />
       </wsdl:message>
       <wsdl:message name="CreateDqsResponse">
       <wsdl:part name="CreateDqsResponse" element="CreateDqsResponse" />
       </wsdl:message>

<wsdl:portType name="WsResourceFactoryPortType" >

   <wsdl:operation name="CreateDqs">
   <wsdl:input  wsa:Action="http://ws.apache.org/muse/test/wsrf/dqs/CreateDqs"
name="CreateDqsRequest" message="tns:CreateDqsRequest" />
   <wsdl:output
wsa:Action="http://ws.apache.org/muse/test/wsrf/dqs/CreateDqsResponse"
name="CreateDqsResponse" message="tns:CreateDqsResponse" />
   </wsdl:operation>

</wsdl:portType>

<wsdl:binding                                          name="WsResourceFactoryBinding"
type="tns:WsResourceFactoryPortType">
<wsdl-soap:binding
 style="document" transport="http://schemas.xmlsoap.org/soap/http" />
</wsdl:binding>

<wsdl:service name="WsResourceFactoryService">

  <wsdl:port name="WsResourceFactoryPort"
binding="tns:WsResourceFactoryBinding">

    <wsdl-soap:address
location="http://localhost:8080/wsrf/services/WsResourceFactory"/>

  </wsdl:port>
```

```
</wsdl:service>

</wsdl:definitions>
```

Cabe ressaltar que o editor gráfico pode auxiliar, ainda que de forma limitada, na varredura por inconsistências no código WSDL, como por exemplo, ao apontar que uma determinada operação, que venha a possuir parâmetros de entrada e saída já definidos corretamente, não possui mensagens SOAP associadas que façam com que tais parâmetros sejam repassados da origem ao destino. A Figura 20 demonstra um exemplo aplicado ao serviço FS, onde a mensagem de retorno da operação foi excluída propositalmente.

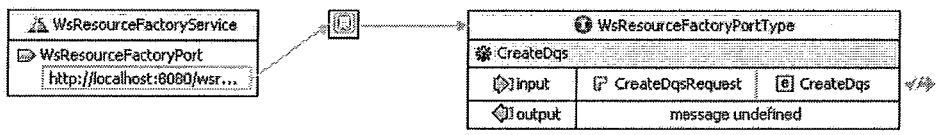

Figura 20 – Erro de mensagem indefinida para operação em WSDL

Uma vez elaborados corretamente os documentos WSDL, o Apache Muse, ao ser inicializado, através do container Tomcat, necessita de um arquivo próprio de configuração denominado *muse.xml*, escrito em linguagem XML e que pode ser editado por linha de comando ou com auxílio do editor do Eclipse TPTP, ou gerado através do uso da ferramenta WSDL2Java (WSDL2JAVA, 2008), embutida no Apache Muse. Este arquivo possui a lista de operações externas expostas às aplicações.

A Figura 21 ilustra o arquivo de configuração do Apache Muse para o GParGRES.

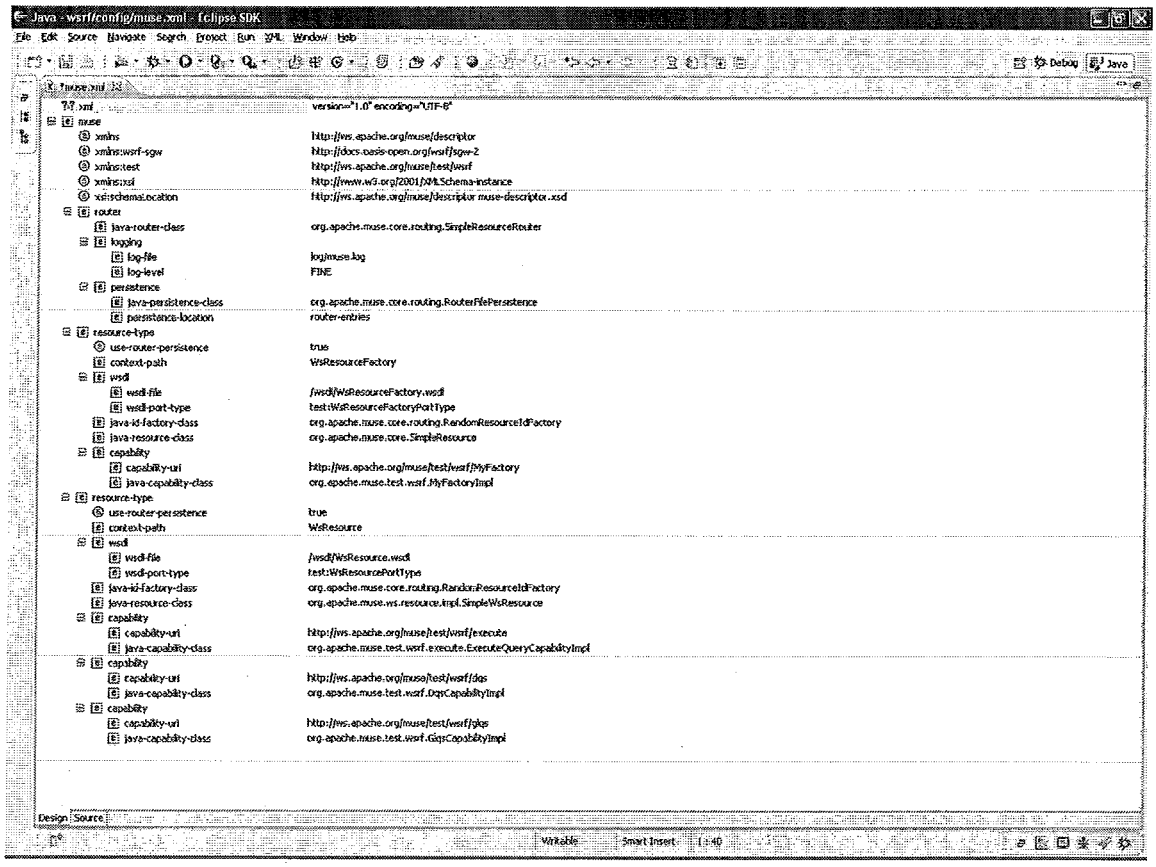

Figura 21 – Arquivo de configuração do Apache Muse

Para o caso de edição manual deste arquivo, não há auxílio por parte do editor para a elaboração deste arquivo de configuração. Eventuais problemas no mesmo só serão detectados no momento de inicialização do servidor hospedeiro Tomcat, uma vez que o próprio Apache Muse irá validar o mesmo em tal etapa.

3.3.2 – Processamento da Consulta e Fragmentação Virtual

O GParGRES não possui um analisador sintático próprio (*parser*), pois este não é um de seus objetivos e, portanto, as consultas são entregues ao GParGRES em um formato pré-determinado, que chamaremos de "consultas parametrizadas".

Estas consultas parametrizadas possuem como particularidade em relação às consultas oferecidas pelo TPC-H o seguinte aspecto: para cada consulta utilizada, são anexados predicados que contém intervalos referentes aos denominados atributos de

fragmentação. Estes atributos são referentes às tabelas de fato, *orders* e *lineitem*, e são respectivamente os campos *o_orderkey* e *l_orderkey*.

A seguir apresentamos como exemplo a consulta Q1 do *benchmark* TPC-H, em sua forma original.

Consulta Q1 TPC-H:

```
select
        l_returnflag,
        l_linestatus,
        sum(l_quantity) as sum_qty,
        sum(l_extendedprice) as sum_base_price,
        sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
        sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as
sum_charge,
        avg(l_quantity) as avg_qty,
        avg(l_extendedprice) as avg_price,
        avg(l_discount) as avg_disc,
        count(*) as count_order
from
        lineitem
where
        l_shipdate <= date '1998-12-01' - interval '90 day'
group by
        l_returnflag,
        l_linestatus
order by
        l_returnflag,
        l_linestatus;
```

Conforme cada umas destas tabelas seja referenciada pela consulta, são adicionados os respectivos predicados com os atributos de fragmentação pertinentes. Além disso, adotamos a reescrita da operação de média (*avg*) também utilizada pelo ParGRES, onde a mesma é substituída por operações de soma (*sum*) e enumeração (*count*) adotados também no ParGRES (LIMA, 2004; LIMA, MATTOSO et al., 2004; MATTOSO, ZIMBRÃO et al., 2005c; MATTOSO, ZIMBRÃO et al., 2005b; MATTOSO, ZIMBRÃO et al., 2005a).

A seguir, apresentamos um exemplo da consulta Q1 na forma como o GParGRES a espera, já reescrita. Os campos em negrito demonstram a reescrita da

operação de média (*avg*) e a presença do atributo de fragmentação *l_orderkey*, devido a referência a tabela *lineitem*. Neste momento, ainda não temos os valores definidos para o atributo de fragmentação, valores que serão preenchidos de acordo com parâmetros citados anteriormente.

Consulta Q1 parametrizada:

```
select
        l_returnflag, l_linestatus,
        sum(l_quantity) as sum_qty,
        sum(l_extendedprice) as sum_base_price,
        sum(l_extendedprice * (1-l_discount)) as sum_disc_price,
        sum(l_extendedprice * (1-l_discount) * (1+l_tax)) as sum_charge,
        sum(l_quantity) as avg_qty_1, count(l_quantity) as avg_qty_2,
        sum(l_extendedprice) as price_1, count(l_extendedprice) as
price_2,
        sum(l_discount) as avg_disc_1, count(l_discount) as avg_disc_2,
        count(*) as count_order
from
        lineitem
where
        l_shipdate <= date '1998-12-01' - interval '90 day'
        and l_orderkey >= ? and l_orderkey < ?
group by
        l_returnflag, l_linestatus
order by
        l_returnflag, l_linestatus;
```

Estes passos foram realizados por não possuirmos um analisador sintático próprio, o que não faz parte dos objetivos do GParGRES. Ressaltamos que o ParGRES possui um analisador e interpretador próprio das consultas a ele repassadas, que as reescreve em tempo de execução para, a partir da cardinalidade dos atributos de fragmentação utilizados pela consulta, calcular os intervalos de valores por eles utilizados e então repassá-los para posterior execução da consultas frente ao SGBD. Portanto, cada consulta, mesmo que acompanhada de intervalos de seleção relativos aos predicados de fragmentação, é reescrita por completo e os intervalos informados pelo usuário são substituídos pelos gerados pelo ParGRES.

Isto demanda atenção especial, pois se cada instância do ParGRES reescrevesse cada sub-consulta por completo, não teríamos o paralelismo desejado. Por este motivo,

foi necessário realizar adaptações no controlador de acesso JDBC do ParGRES, que passou a receber os valores mínimo e máximo dos atributos de fragmentação, quando aplicáveis.

Também foi necessário inibir a reescrita da consulta, para que o ParGRES não calculasse novamente os intervalos de predicados, e sim respeitasse os valores repassados pelo GParGRES. Apesar da necessidade das modificações, não foi preciso alterar os mecanismos de balanceamento de carga e composição de resultados do ParGRES, que continuaram a operar da forma original, de forma que continuamos a obter os benefícios da AVP (LIMA, 2004) nos agrupamentos da grade.

Os valores dos intervalos dos atributos de fragmentação, em negrito na consulta anteriormente apresentada, não são informados diretamente na consulta, pois são calculados durante o processo de fragmentação virtual simples – SVP - (AKAL, BÖHM et al., 2002) adotado pelo GParGRES, com o uso dos parâmetros fator de escala (*scale_factor*) e da quantidade de serviços GLQS instanciados (*glqs_list*), obtidos através do arquivo de configuração do GParGRES e apresentados na seção 3.2 –. Os atributos de fragmentação receberão como valores possíveis o resultado da seguinte regra de formação:

valor mínimo para atributo = 1
*valor máximo para atributo = fator de escala * 6.000.000*

Esta operação, realizada no componente DQS, é responsável por, a partir deste cálculo, gerar sub-consultas e repassá-las a cada GLQS a ser criado, que por sua vez deve então orientar o ParGRES a executar a sub-consulta apenas sob o intervalo selecionado.

Para ilustrar o processo de substituição dos valores do intervalo do atributo de fragmentação no exemplo citado nesta seção, apresentamos as sub-consultas geradas pela técnica de fragmentação virtual simples adotada pelo GParGRES, supondo que através do arquivo de configuração do GParGRES, obtivemos o fator de escala do *benchmark* TPC-H igual a 1, o que gera atributos de fragmentação com valores possíveis entre 1 e 6.000.000, e que a lista de GLQS informa ao GParGRES que crie 2 GLQS. Dessa forma, a consulta parametrizada será reescrita em duas sub-consultas, cada qual a ser repassada para um GLQS, conforme apresentado a seguir. O trecho em negrito destaca a reescrita dos valores dos atributos de fragmentação.

sub-consulta Q1a:

```
select
        l_returnflag, l_linestatus,
        sum(l_quantity) as sum_qty,
        sum(l_extendedprice) as sum_base_price,
        sum(l_extendedprice * (1-l_discount)) as sum_disc_price,
        sum(l_extendedprice * (1-l_discount) * (1+l_tax)) as sum_charge,
        sum(l_quantity) as avg_qty_1, count(l_quantity) as avg_qty_2,
        sum(l_extendedprice) as price_1, count(l_extendedprice) as
price_2,
        sum(l_discount) as avg_disc_1, count(l_discount) as avg_disc_2,
        count(*) as count_order
from
        lineitem
where
        l_shipdate <= date '1998-12-01' - interval '90 day'
        and l_orderkey >= 1 and l_orderkey < 3000001
group by
        l_returnflag, l_linestatus
order by
        l_returnflag, l_linestatus;
```

sub-consulta Q1b:

```
select
        l_returnflag, l_linestatus,
        sum(l_quantity) as sum_qty,
        sum(l_extendedprice) as sum_base_price,
        sum(l_extendedprice * (1-l_discount)) as sum_disc_price,
        sum(l_extendedprice * (1-l_discount) * (1+l_tax)) as sum_charge,
        sum(l_quantity) as avg_qty_1, count(l_quantity) as avg_qty_2,
        sum(l_extendedprice) as price_1, count(l_extendedprice) as
price_2,
        sum(l_discount) as avg_disc_1, count(l_discount) as avg_disc_2,
        count(*) as count_order
from
        lineitem
where
        l_shipdate <= date '1998-12-01' - interval '90 day'
        and l_orderkey >= 3000001 and l_orderkey < 6000001
group by
        l_returnflag, l_linestatus
```

```
order by
        l_returnflag, l_linestatus;
```

A seguir apresentamos o processo de execução local da consulta, realizado por cada instância de GLQS criada pelo DQS.

3.3.3 – Execução Local da Consulta

A execução local da consulta é realizada através do serviço GLQS e uma operação capacitadora (*capability*) específica (não exposta ao mundo externo) que tem por objetivo conectar-se ao ParGRES associado via controlador de acesso JDBC do mesmo e solicitar a execução da sub-consulta repassada pelo DQS a cada instância de GLQS.

Conforme especificado na seção 3.3.1 –, são repassados ao ParGRES: a sub-consulta com os intervalos dos atributos de fragmentação, os valores mínimos e máximos dos mesmos, para auxiliar na execução da consulta.

Cabe ressaltar que a coordenação da execução das sub-consultas, o que compreende a: solicitação de execução de cada uma das mesmas pelo DQS aos GLQS pertinentes; o monitoramento do estado de execução de cada GLQS pelo DQS para verificação da finalização ou não da execução de cada sub-consulta; e os passos associados à composição de resultados.

Toda a coordenação da consulta é realizada pelo DQS, que usa de recursos multi-tarefa da linguagem Java para solicitar tanto a criação dos serviços GLQS quanto o disparo em paralelo das diversas submissões de consulta, cada qual para seu respectivo GLQS. A composição dos resultados é explicitada na seção a seguir.

3.3.4 – Composição de Resultados

A composição de resultados do GParGRES segue os mesmos princípios do ParGRES, com o uso de uma base de dados temporária, criada em memória, através do uso do SGBD HSQLDB (HSQLDB, 2008). O DQS utiliza um dos parâmetros do arquivo XML de configuração do GParGRES para extrair o comando de criação da tabela na referida base de dados em memória (*CREATE TABLE*).

A seguir, apresentaremos um exemplo de composição de resultados com base na consulta Q5 do *benchmark* TPC-H.

Consulta Q5 (TPC-H):

```
select
        n_name, sum(l_extendedprice * (1 - l_discount)) as revenue
from
        customer, orders, lineitem, supplier, nation, region
where
        c_custkey = o_custkey and l_orderkey = o_orderkey
        and l_suppkey = s_suppkey and c_nationkey = n_nationkey
        and s_nationkey = n_nationkey and n_regionkey = r_regionkey
        and r_name = 'ASIA'
        and o_orderdate >= date '1994-01-01'
        and o_orderdate < date '1994-01-01' + interval '1 year'
group by
        n_name
order by
        revenue desc;
```

Esta consulta acessa as duas tabelas de fato (*orders* e *lineitem*) e demais tabelas de dimensão. Para a montagem do comando de criação de tabela temporária em memória, devemos criar colunas que contenham o tipo de dados esperado pelos componentes da cláusula SELECT na consulta original. No caso da consulta em questão, teremos o seguinte comando para criação de tabela:

Comando para criação de tabela (consulta Q5):

```
create table RC
        (n_name CHAR(25), revenue DECIMAL);
```

O objetivo de tal tabela é receber as inserções resultados provenientes da execução de cada sub-consulta associada a um GLQS. A criação desta tabela, e consequentemente da base de dados em memória para contê-la, ocorre no recurso da grade que contém a instância de DQS.

A tabela é criada com o nome RC no exemplo, associado à composição de resultados (*resultcomposer*), mas tal nome é facultativo ao usuário, pois não é requerido pelo GParGRES. Apenas deve-se utilizar o mesmo nome para a criação da tabela assim como para os comandos de inserção e recuperação de dados posteriores.

Como apresentado nesta seção, cada GLQS é responsável por solicitar a execução de uma sub-consulta ao ParGRES e conseqüentemente, recebe do mesmo o resultado esperado para a consulta em questão. Ao receber tal resultado, cada GLQS gera uma série de assertivas de inserção de dados que serão repassadas ao DQS que o criou, para que os resultados obtidos sejam então inseridos pelo DQS em sua tabela temporária. Ao final, o DQS terá então uma lista com assertivas de inserção, próprias a consulta em questão, que ilustramos a seguir para a consulta Q5 do *benchmark* TPC-H.

Comandos de inserção (consulta Q5):

```
insert into RC values ('CHINA',7822103.0000);
insert into RC values ('INDIA', 6376121.5085);
insert into RC values ('JAPAN', 6000077.2184);
insert into RC values ('INDONESIA', 5580475.4027);
insert into RC values ('VIETNAM', 4497840.5466);
```

Após o término da execução de todas as sub-consultas entregues aos GLQS, e a respectiva inserção dos dados parciais na tabela temporária da base de dados criada em memória pelo DQS, dá-se início a recuperação dos dados com o comando SELECT adequado para a consulta em questão e informado através de um arquivo referenciado como parâmetro recebido pelo GParGRES, conforme citado na seção 3.2 –.

A seguir apresentamos o comando de recuperação de dados associado à consulta Q5 do *benchmark* TPC-H. Este comando terá como função coletar o resultado esperado pela consulta submetida originalmente ao GParGRES, que estará armazenado na tabela temporária em memória, após o processo de inserção de dados na mesma pelo DQS.

Comando para recuperação de dados (consulta Q5):

```
select
        n_name, sum(revenue) as final_revenue
from RC
group by
        n_name
order by
        final_revenue desc;
```

Portanto, para cada consulta submetida ao GParGRES, teremos então um par de comandos de criação de tabela e recuperação de dados, além de um conjunto de assertivas de inserção de dados, estas últimas montadas em tempo real, durante a execução de cada sub-consulta em cada GLQS.

Cabem alguns comentários sobre o processo de composição resultados como um todo, desde o momento de criação da tabela temporária até a recuperação dos dados. Em primeiro lugar, a tabela temporária é criada no recurso da grade (servidor) que possui a instância do DQS associada, para evitar o uso de área de memória adicional nos recursos que executam instâncias de GLQS ou do próprio ParGRES. Estes recursos demandam a maior capacidade de processamento, memória e disco possíveis, pois sua função deve concentrar-se em garantir a execução das sub-consultas com alto desempenho;

Adicionalmente, ao optar por um ponto pré-determinado para a inserção dos dados (o recurso com a instância de DQS), se ganha em facilidade para gerenciar a posterior inserção de dados provenientes de cada sub-consulta executada por uma instância de GLQS, uma vez que somente será preciso gerenciar uma única tabela em uma única base de dados em memória (no próprio recurso com o DQS) e também obtemos facilidade na recuperação de dados através do comando de recuperação de dados final, pela mesma razão apresentada.

3.4 – Diagrama Comparativo

Uma vez listados os padrões, produtos e alternativas para processamento de consultas em ambientes de grade, assim como o GParGRES, seus serviços, sua arquitetura e seus processos, apresentamos um diagrama que concentra funcionalidades relevantes para esta dissertação, observadas entre todas as soluções para bancos de dados e grades e o GParGRES.

Nosso objetivo é apontar pontos favoráveis e pertinentes a cada uma das mesmas, apresentar similaridades e diferenças de forma que possamos montar uma estrutura comparativa entre as soluções estudadas e o GParGRES.

A Tabela 5 apresenta a consolidação de tais pontos, de acordo com características que consideramos relevantes para o problema abordado por esta dissertação.

Tabela 5 – Quadro Comparativo de Soluções

Solução / Característica	Heterogeneidade do Ambiente	Paralelismo Inter-/ Intra-consulta	Técnica de Desenvolvimento	Nível de Generalidade da Solução
OGSA-DAI	Heterogêneo	Não oferece suporte	Serviço *web*	Médio
OGSA-DQP	Heterogêneo	Oferece suporte com restrições	Serviço *web*	Médio
Spitfire	Heterogêneo	Não oferece suporte	Serviço *web*	Alto
Oracle 11g	Tende a homogêneo	Oferece suporte	Aplicação proprietária – código fechado	Baixo
Grade habilitada para OLAP *	Heterogêneo	Oferece suporte com restrições	Serviço *web*	Baixo
Armazéns de Dados em grades**	Heterogêneo	Oferece suporte com restrições	Manipulação direta da base de dados – particionamento e indexação via árvores-X	Baixo
GParGRES	Heterogêneo	Oferece suporte	Serviço para grade	Alto

* - solução criada por (LAWRENCE, DEHNE et al., 2007; LAWRENCE e RAU-CHAPLIN, 2006).

** - solução criada por (WERHLE, MIQUEL et al., 2005).

Cabe também realizarmos alguns comentários a respeito da característica denominada "Nível de generalidade da solução". Por este conceito queremos dizer o quão independente de fatores como estruturas especiais de dados, mediadores de grades, operações sobre as bases de dados, como fragmentação física, por exemplo, cada aplicação é caracterizada.

Um nível baixo de generalização representa que para utilizar uma solução, é necessário elaborar um ambiente personalizado e distante de padrões para

desenvolvimento de soluções para grades, além de requer estruturas de dados específicas para seu funcionamento, por exemplo. Já um nível estabelecido como médio apresenta como característica a adoção de padrões para desenvolvimento de serviços para grade, mas estes padrões foram implementados e encapsulados em mediadores de grade específicos, como no caso da OGSA-DAI e OGSA-DQP e o respectivo Globus WSRF (GLOBUS WSRF, 2008).

O nível alto de generalidade assegura que a solução segue padrões estabelecidos e implementados independente de mediadores de grades, estruturas de dados especiais ou necessidade de fragmentação física da base de dados.

Capítulo 4 – Avaliação do GParGRES

4.1 – Configuração do Ambiente

A seguir são apresentadas as características relacionadas à montagem do ambiente utilizado na realização dos experimentos desta dissertação. Inicialmente, descrevemos como o *benchmark* TPC-H foi configurado, quais parâmetros foram adotados para a criação de sua base de dados e características pertinentes à mesma.

Em seguida, ilustramos as consultas selecionadas dentre o lote disponibilizado pelo TPC-H. Descrevemos as características de cada uma das mesmas de forma sumarizada.

Posteriormente, enumeramos os cenários para a realização dos experimentos e apresentamos os resultados obtidos. Todos os experimentos foram realizados com auxílio das ferramentas fornecidas pelo *Grid'5000* para agendamento de tarefas e criação de ambientes de sistema personalizados.

4.2 – Utilização do Benchmark TPC-H

4.2.1 – Configuração Geral

Para realizarmos a execução dos experimentos com o GParGRES, adotamos o uso do *benchmark* TPC-H, que ilustra uma base de dados e consultas que refletem aplicações tipicamente OLAP. Na seção 2.5 – apresentamos a definição do *benchmark* em questão e o respectivo esquema da base de dados, com detalhes referentes à criação da mesma.

Uma vez criada a base de dados, replicamos a mesma em cada um dos recursos pertencentes a cada um dos agrupamentos que compõem os nós da grade utilizados para os experimentos.

As duas tabelas de fato, *orders* e *lineitem*, representam aproximadamente 80% do total de tuplas da base de dados e foram fisicamente ordenadas de acordo com os atributos de fragmentação das mesmas (o_*orderkey* para a tabela *orders* e l_*orderkey* para a tabela *lineitem*). Estes atributos possuem cardinalidade definida de acordo com o intervalo [1 .. SF * 6.000.000], onde SF é igual ao fator de escala utilizado na criação da base de dados do *benchmark* TPC-H.

Foram construídos índices com base nos atributos de fragmentação e também foram criados índices para cada uma das chaves estrangeiras de todas as tabelas da base. Finalmente, foram atualizadas as estatísticas da base de dados para uso das mesmas pelo otimizador do próprio SGBD.

Após estas operações, o tamanho total da base de dados, incluindo índices, alcançou o valor aproximado de 2,2GBytes, considerando que o fator de escala adotado foi igual a 1.

4.2.2 – Consultas

Dentre o lote de consultas disponibilizadas para uso com o *benchmark* TPC-H, optamos por utilizar as consultas Q1, Q5, Q6, Q12, Q14 e Q18, que são descritas a seguir. As consultas foram geradas através do aplicativo *qgen*, parte componente do TPC-H, conforme orientação do próprio *benchmark*.

Consulta Q1:

```
select
        l_returnflag,
        l_linestatus,
        sum(l_quantity) as sum_qty,
        sum(l_extendedprice) as sum_base_price,
        sum(l_extendedprice * (1-l_discount)) as sum_disc_price,
        sum(l_extendedprice * (1-l_discount) * (1+l_tax)) as sum_charge,
        avg(l_quantity) as avg_qty,
        avg(l_extendedprice) as avg_price,
        avg(l_discount) as avg_disc,
        count(*) as count_order
from
        lineitem
where
        l_shipdate <= date '1998-12-01' - interval '90 day'
group by
        l_returnflag,
        l_linestatus
order by
        l_returnflag,
        l_linestatus;
```

Consulta Q5:

```
select
        n_name,
        sum(l_extendedprice * (1-l_discount)) as revenue
from
        customer,
        orders,
        lineitem,
        supplier,
        nation,
        region
where
        c_custkey = o_custkey
        and l_orderkey = o_orderkey
        and l_suppkey = s_suppkey
```

```
        and c_nationkey = s_nationkey
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = 'ASIA'
        and o_orderdate >= date '1994-01-01'
        and o_orderdate < date '1994-01-01' + interval '1 year'
group by
        n_name
order by
        revenue desc;
```

Consulta Q6:

```
select
        sum(l_extendedprice * l_discount) as revenue
from
        lineitem
where
        l_shipdate >= date '1994-01-01'
        and l_shipdate < date '1994-01-01' + interval '1 year'
        and l_discount between .06 - 0.01 and .06 + 0.01
        and l_quantity < 24;
```

Consulta Q12:

```
select
        l_shipmode,
        sum(case
                when o_orderpriority = '1-URGENT'
                        or o_orderpriority = '2-HIGH'
                        then 1
                else 0
        end) as high_line_count,
        sum(case
                when o_orderpriority <> '1-URGENT'
                        and o_orderpriority <> '2-HIGH'
                        then 1
                else 0
        end) as low_line_count
from
        orders,
        lineitem
where
        o_orderkey = l_orderkey
        and l_shipmode in ('MAIL', 'SHIP')
        and l_commitdate < l_receiptdate
        and l_shipdate < l_commitdate
        and l_receiptdate >= date '1994-01-01'
        and l_receiptdate < date '1994-01-01' + interval '1 year'
group by
        l_shipmode
order by
        l_shipmode;
```

Consulta Q14:

```
select
        100.00 * sum(case
                when p_type like 'PROMO%'
                        then l_extendedprice * (1-l_discount)
                else 0
        end) / sum(l_extendedprice * (1-l_discount)) as promo_revenue
from
        lineitem,
        part
where
        l_partkey = p_partkey
        and l_shipdate >= date '1995-09-01'
```

```
        and l_shipdate < date '1995-09-01' + interval '1 month';
```

Consulta Q18:

```
select
        c_name,
        c_custkey,
        o_orderkey,
        o_orderdate,
        o_totalprice,
        sum(l_quantity)
from
        customer,
        orders,
        lineitem
where
        o_orderkey in (
                select
                        l_orderkey
                from
                        lineitem
                group by
                        l_orderkey having
                                sum(l_quantity) > 300
        )
        and c_custkey = o_custkey
        and o_orderkey = l_orderkey
group by
        c_name,
        c_custkey,
        o_orderkey,
        o_orderdate,
        o_totalprice
order by
        o_totalprice desc,
        o_orderdate;
```

A seguir, apresentamos algumas das principais características associadas a cada uma das consultas utilizadas na avaliação do GParGRES.

Q1 – opera apenas sobre a tabela *lineitem* e realiza diversas operações de agregação (*sum* e *avg*).

Q5 – executa diversas junções entre praticamente todas as tabelas da base de dados, e apenas uma operação de agregação.

Q6 – similar à consulta Q1. Acessa a tabela *lineitem* e realiza uma única operação de agregação.

Q12 – aplica uma junção entre as tabelas *orders* e *lineitem* e realiza operações de agregação.

Q14 – opera uma junção entre *lineitem* (tabela de fatos) e *part* (tabela de dimensão).

Q18 – realiza junções entre as duas tabelas de fato (*customer* e *lineitem*) e a tabela *customer*.

De forma similar ao observado em (LIMA, 2004), as consultas repassadas para execução ao GParGRES foram reescritas com base nos atributos de fragmentação, o_*orderkey* – tabela *orders* e l_*orderkey* – tabela *lineitem*, que serão posteriormente utilizados pela técnica de paralelismo do GParGRES, para que sejam geradas sub-consultas a serem entregues a cada GLQS. O paralelismo adotado pelo GParGRES, assim como a utilização dos atributos de fragmentação em questão são detalhados na seção 3.3.2 –.

Conforme apresentado na seção 3.3.2 –, o GParGRES recebe como dado de entrada a consulta denominada como "consulta parametrizada", o que corresponde a um processo de reescrita realizado pelo próprio usuário, o que, para os experimentos em questão, corresponde às consultas apresentadas a seguir.

As linhas em negrito representam as mudanças em relação às consultas originais. Estas mudanças são referentes ora à inserção de predicados referentes aos atributos de fragmentação, *l_orderkey* (tabela *lineitem*) e *o_orderkey* (tabela *orders*). Estes atributos podem aparecer isoladamente, quando a consulta referencia apenas a tabela em questão ou simultaneamente, quando a consulta referenciar as duas tabelas.

Também é possível notar a reescrita das operações de média (*avg*) por operações de somatório e contabilização (*sum* e *count*, respectivamente), como adotado em (LIMA, 2004). A reescrita dessas operações foi adotada, pois o GParGRES não possui analisador sintático das consultas (*parser*), uma vez que este não fazia parte do escopo desta dissertação e pela necessidade de informar ao ParGRES as operações citadas da forma descrita, uma vez que inibimos a reescrita da consulta pelo ParGRES, conforme apresentado na seção 3.3.2 –.

Consulta Q1:

```
select l_returnflag, l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1-l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1-l_discount) * (1+l_tax)) as sum_charge,
    sum(l_quantity) as avg_qty_1, count(l_quantity) as avg_qty_2,
    sum(l_extendedprice) as price_1, count(l_extendedprice) as price_2,
    sum(l_discount) as avg_disc_1, count(l_discount) as avg_disc_2,
    count(*) as count_order
from
    lineitem
```

```
where
    l_shipdate <= date '1998-12-01' - interval '90 day'
    and l_orderkey > :Ii and l_orderkey <= :If
group by
    l_returnflag, l_linestatus
order by
    l_returnflag, l_linestatus;
```

Consulta Q5:

```
select
      n_name, sum(l_extendedprice * (1 - l_discount)) as revenue
from
      customer,
      orders,
      lineitem,
      supplier,
      nation,
      region
where
      c_custkey = o_custkey
      and l_orderkey = o_orderkey
      and l_suppkey = s_suppkey
      and c_nationkey = n_nationkey
      and s_nationkey = n_nationkey
      and n_regionkey = r_regionkey
      and r_name = 'ASIA'
      and o_orderdate >= date '1994-01-01'
      and o_orderdate < date '1994-01-01' + interval '1 year'
      and o_orderkey >= :Ii and o_orderkey < :If
      and l_orderkey >= :Ii and l_orderkey < :If
group by
       n_name
order by
       revenue desc;
```

Consulta Q6:

```
select
      sum(l_extendedprice * l_discount) as revenue
from
      lineitem
where
      l_shipdate >= date '1994-01-01'
      and l_shipdate < date '1994-01-01' + interval '1 year'
      and l_discount between .06 - 0.01 and .06 + 0.01
      and l_quantity < 24
      and l_orderkey >= :Ii and l_orderkey < :If;
```

Consulta Q12:

```
select
      l_shipmode,
      sum ( case when o_orderpriority = '1-URGENT' or
                      o_orderpriority = '2-HIGH'
            then 1 else 0
            end
          ) as high_line_count,
      sum ( case when o_orderpriority <> '1-URGENT' and
                      o_orderpriority <> '2-HIGH'
            then 1 else 0
            end
          ) as low_line_count
from
      orders,
      lineitem
where
      o_orderkey = l_orderkey
```

```
        and l_shipmode in ('MAIL', 'SHIP')
        and l_commitdate < l_receiptdate
        and l_shipdate < l_commitdate
        and l_receiptdate >= date '1994-01-01'
        and l_receiptdate < date '1994-01-01' + interval '1 year'
        and o_orderkey >= :Ii and o_orderkey < :If
        and l_orderkey >= :Ii and l_orderkey < :If
group by
        l_shipmode
order by
        l_shipmode;
```

Consulta Q14:

```
select
        100.00 * sum( case when p_type like 'PROMO%'
                        then l_extendedprice * (1-l_discount)
                        else 0
                        end
                      ) as promo_revenue_1,
        sum(l_extendedprice * (1-l_discount)) as promo_revenue_2
from
        lineitem,
        part
where
        l_partkey = p_partkey and l_shipdate >= date '1995-09-01'
        and l_shipdate < date '1995-09-01' + interval '1 month'
        and l_orderkey >= :Ii and l_orderkey < :If;
```

Consulta Q18:

```
select
        c_name,
        c_custkey,
        o_orderkey,
        o_orderdate,
        o_totalprice,
        sum(l_quantity)
from
        customer,
        orders,
        lineitem
where
        o_orderkey in (
                select
                        l_orderkey
                from
                        lineitem
                where l_orderkey >= :Ii and l_orderkey < :If
                group by
                        l_orderkey having
                                sum(l_quantity) > 300
        )
        and c_custkey = o_custkey
        and o_orderkey = l_orderkey
        where o_orderkey >= :Ii and o_orderkey < :If
        where l_orderkey >= :Ii and l_orderkey < :If
group by
        c_name,
        c_custkey,
        o_orderkey,
        o_orderdate,
        o_totalprice
order by
        o_totalprice desc,
        o_orderdate;
```

Os parâmetros **:Ii** e **:If** são substituídos em tempo de execução pelos intervalos de menor e maior valor de cada atributo de fragmentação, de acordo com a técnica de fragmentação virtual simples efetuada pelo GParGRES. Relembramos que esta operação utiliza como base dois parâmetros informados no arquivo de configuração do GParGRES (fator de escala - *scale_factor* e a lista de serviços GLQS - *glqs_list*).

Dessa forma, cada consulta submetida ao GParGRES é utilizada como base pelo serviço DQS para que então sejam criadas sub-consultas que serão repassadas aos devidos GLQS. Na seção 3.3.2 –, apresentamos um exemplo de criação de sub-consultas com base no fator de escala que foi utilizado para a realização dos experimentos do GParGRES, supondo a existência de duas instâncias de serviços GLQS.

4.3 – Cenários dos Experimentos

Nesta seção apresentamos os cenários utilizados para a execução dos experimentos, que compõem a base para a avaliação posterior dos resultados observados. Inicialmente, foram escolhidos três cenários principais para os testes. A seguir apresentamos cada um dos mesmos e listamos o objetivo relacionado para com cada um dos cenários.

O primeiro cenário conta com a execução do GParGRES no agrupamento parasol, de menor capacidade de recursos físicos disponível no sítio de Rennes.

O segundo cenário envolve a utilização do GParGRES no agrupamento paraquad, com o dobro de capacidade de memória RAM, e de características de recursos físicos (fabricante, CPU, disco, quantidade de nós) heterogêneas em relação ao agrupamento parasol. O terceiro cenário compreende o uso do GParGRES entre os dois agrupamentos distintos.

O GParGREs foi implementado com o conjunto Apache Muse 2.2.0, Tomcat5.5 e linguagem Java 1.5 e os nós de cada agrupamento têm como SGBD utilizado o PostgreSQL 8.2.4.

Para todos os cenários foi utilizada a base de dados do *benchmark* TPC-H com fator de escala igual a 1, o que gera uma base de aproximadamente 2,2GBytes e as consultas executadas foram: Q1, Q5, Q6, Q12, Q14 e Q18. Cada consulta foi executada dez vezes e para efeito de cálculo, foi descartada a primeira execução de cada uma das consultas, devido ao *cache* frio.

O tempo contabilizado conta também com o tempo necessário para troca de mensagens entre os serviços da arquitetura GParGRES. O *link* que interliga os agrupamentos em cada sítio tem a capacidade de 1Gbps.

Apresentamos uma pequena descrição de cada um dos agrupamentos envolvidos nos experimentos na Tabela 6.

Tabela 6 – Agrupamentos utilizados para os experimentos

Agrupamento	parasol	paraquad
Sítio	Rennes	Rennes
Nós	64	64
Modelo	Sun Fire V20z	Dell PowerEdge 1950
CPU	AMD Opteron 248 - 2.2GHz	Intel Xeon 5148 LV - 2.33GHz
Memória	2GB	4GB
Disco	73GB	160GB
Rede	Gigabit Ethernet	Gigabit Ethernet

As curvas observadas na figuras a seguir demonstram o tempo para execução de cada consulta, em milisegundos. Cada ponto destacado representa o resultado obtido com a quantidade associada de recursos utilizados. O objetivo dos experimentos foi avaliar se o GParGRES apresentaria resultado com aceleração próxima à obtida pelo ParGRES, sem que houvesse prejuízo no tempo observado, o que foi observado com sucesso.

4.3.1 – GParGRES – Agrupamento *Parasol*

O primeiro conjunto de experimentos foi realizado no agrupamento parasol, que dentre os agrupamentos em atividade em Rennes, é aquele que possui menor capacidade de processamento, memória e disco. A seguir, apresentamos o resultado da execução das consultas do *benchmark* TPC-H, na Figura 22, em escala de milisegundos.

Na sequência, também apresentamos o gráfico da execução dos experimentos em escala logarítmica, na Figura 23. Em todas as repetições, o componente GLQS foi alocado na mesma máquina do CQP do ParGRES e o componente DQS foi instanciado em outro recurso do próprio agrupamento, juntamente com os componentes FS e RS e à parte de quaisquer outros componentes do ParGRES.

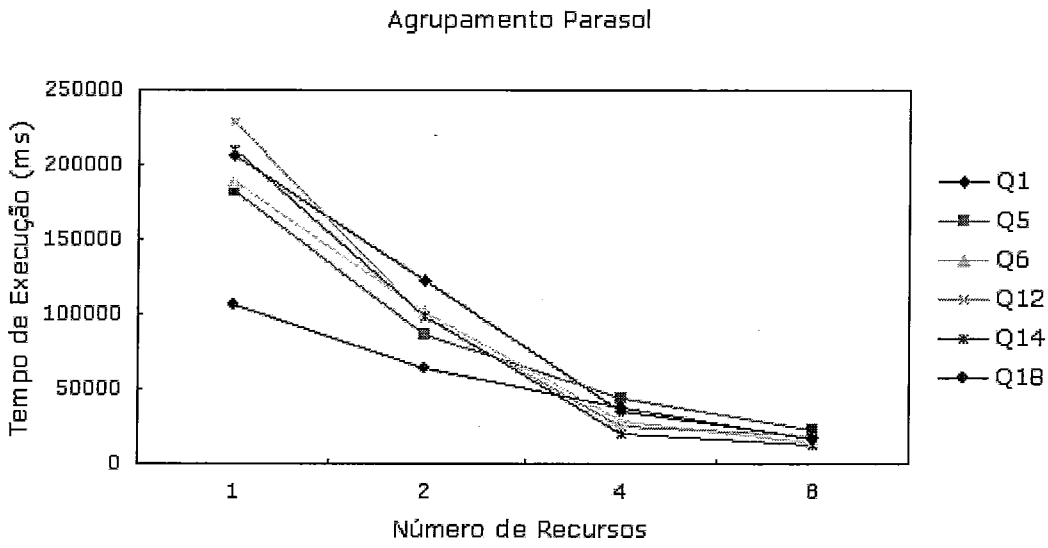

Figura 22 – Execução do GParGRES no agrupamento parasol

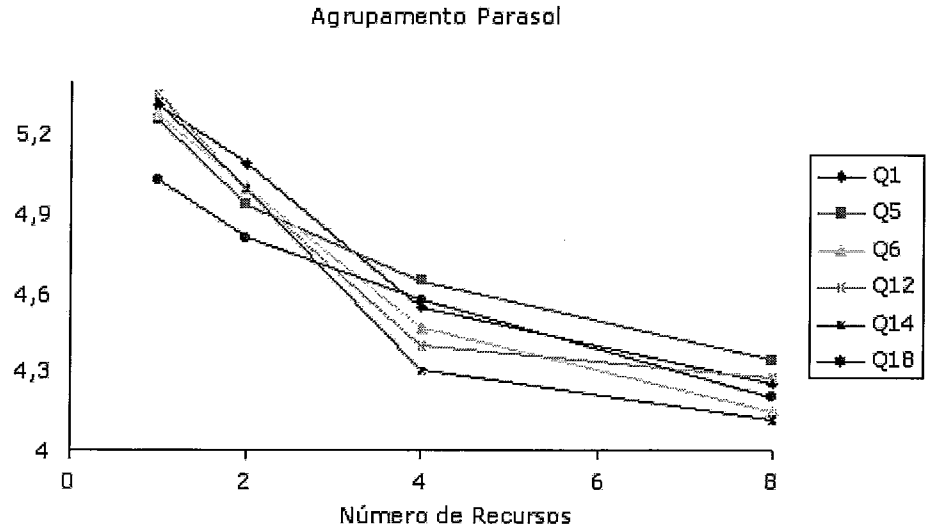

Figura 23 – Gráfico Logarítmico do Agrupamento Parasol

4.3.2 – GParGRES – Agrupamento Paraquad

O segundo cenário de experimentos concentra a execução da aplicação GParGRES no agrupamento paraquad. Foi utilizada a mesma abordagem do cenário no anterior para alocação de componentes das arquiteturas envolvidas. Os resultados são apresentados na Figura 24 em escala de tempo (milisegundos) e na Figura 25, em escala logarítmica.

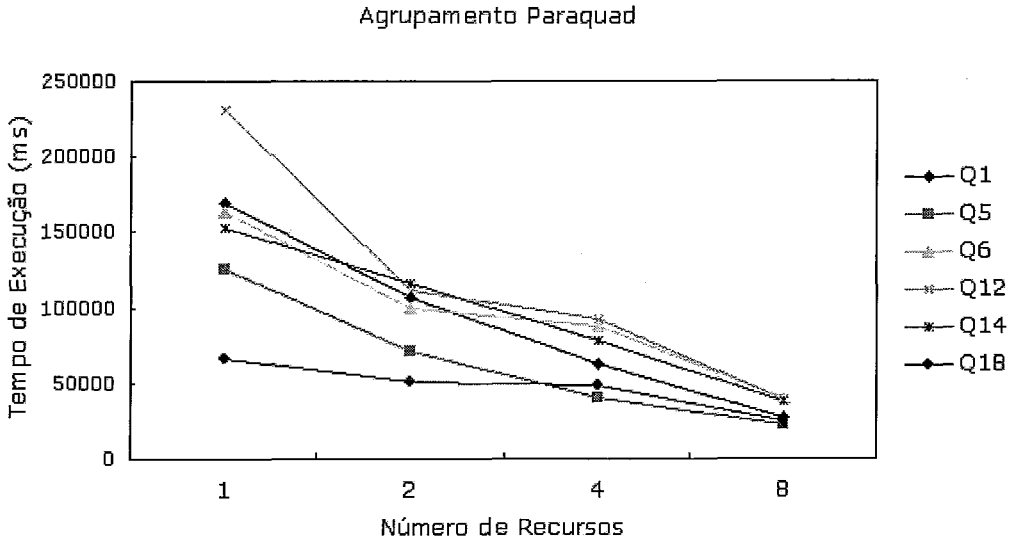

Figura 24 – Execução do GParGRES no agrupamento paraquad

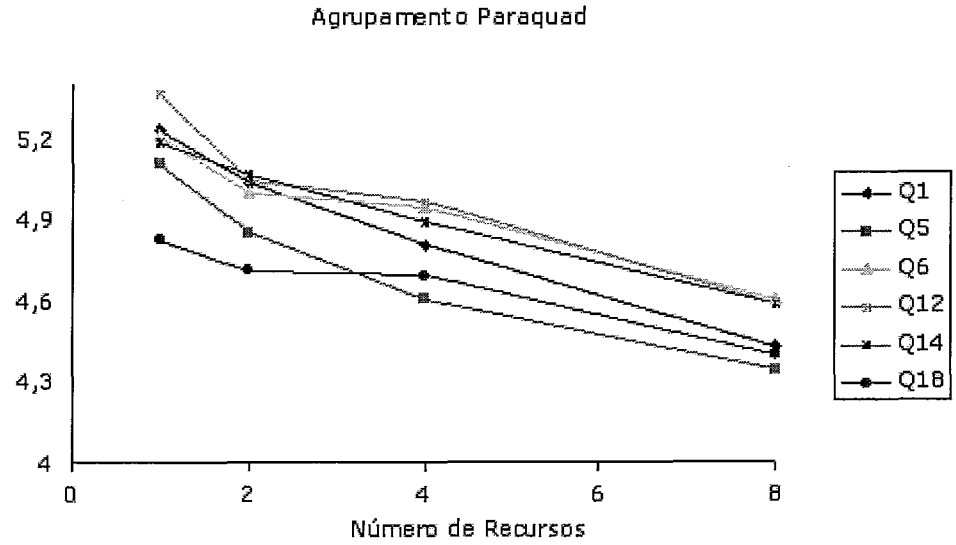

Figura 25 – Gráfico Logarítmico do Agrupamento Paraquad

4.3.3 – GParGRES – Agrupamentos Parasol e Paraquad

O terceiro cenário de realização de experimentos envolveu a utilização de dois agrupamentos distintos, parasol e paraquad. Esta configuração recebeu o nome de modo de execução compartilhada. Deve-se ressaltar que para este cenário, a execução de testes com um nó foi concentrada no agrupamento paraquad, de melhor desempenho teórico, e que as demais configurações contaram com o número igual de recursos divididos entre cada um dos agrupamentos utilizados. Portanto, para dois recursos utilizados no experimento, um foi alocado no agrupamento parasol enquanto o outro no paraquad, e assim por diante.

Cada agrupamento conta com seu próprio GLQS, associado à mesma máquina onde o componente CQP do ParGRES está alocado. Os serviços FS, RS e DQS foram instanciados no agrupamento parasol, em uma máquina à parte dos demais componentes do ParGRES. A escolha por tal localização teve por objetivo não influenciar no resultado obtido, uma vez que dessa forma não haveria processamento concorrente ao ParGRES ou a outros serviços da arquitetura GParGRES. Neste cenário, temos a execução da consulta compartilhada entre recursos heterogêneos e geograficamente distribuídos na grade, o que caracteriza um dos possíveis cenários desejados para a execução de experimentos com o GParGRES. Os resultados são apresentados na Figura 26 e na Figura 27, em escala de tempo (milisegundos) e logarítmica, respectivamente.

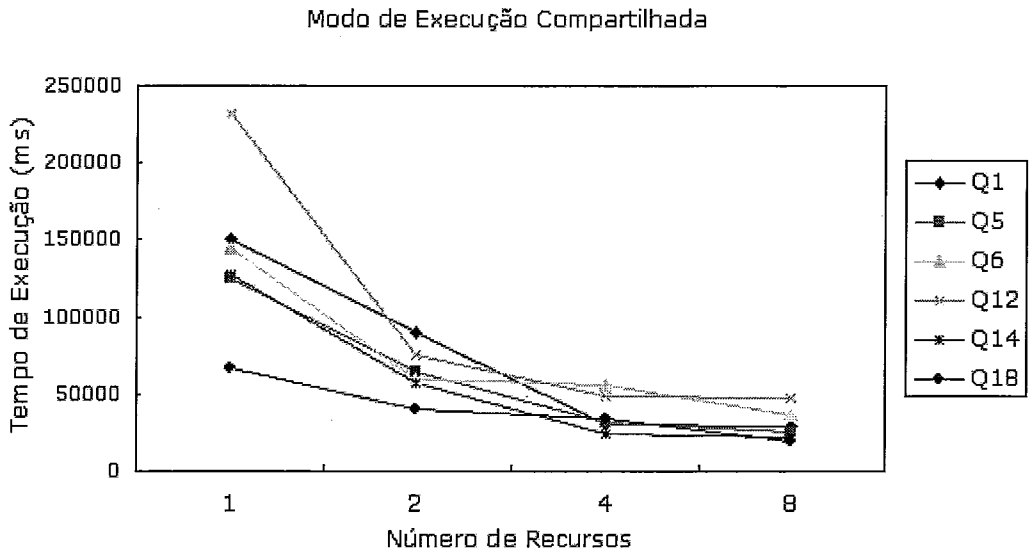

Figura 26 – Execução do GParGRES em modo compartilhado

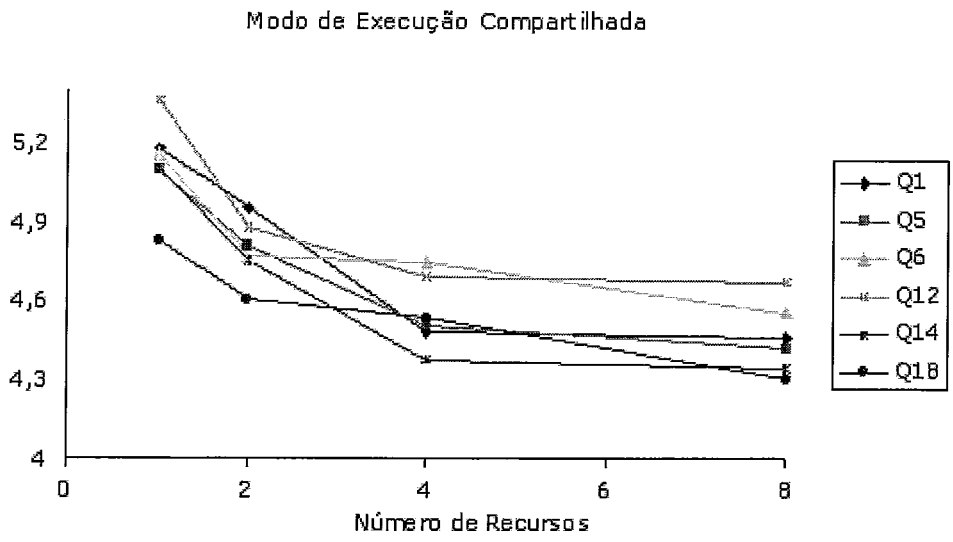

Figura 27 – Gráfico Logarítmico do Modo Compartilhado

4.4 – Avaliação dos Resultados

Quando observamos os resultados referentes ao processamento de um nó apenas, o agrupamento paraquad apresenta valores melhores frente ao parasol devido à sua capacidade de recursos físicos superior. Por outro lado, conforme adicionamos mais recursos (o que aumenta a probabilidade de encontrar fragmentos virtuais em memória), o resultado de ambos os agrupamentos tendem a aproximar-se. Isto representa um indício da eficácia da abordagem não intrusiva do ParGRES e conseqüentemente, do GParGRES. Na Figura 28 e na Figura 29 apresentamos a consolidação dos resultados obtidos para cada cenário de experimento realizado, em escala temporal e logarítmica, respectivamente.

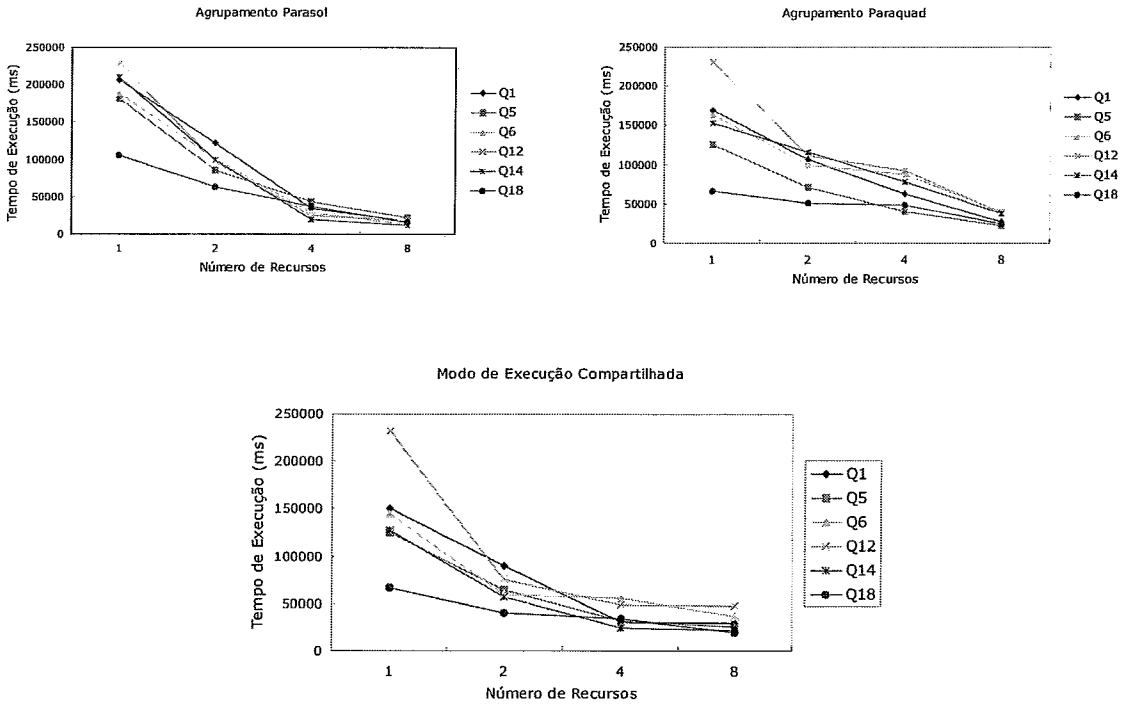

Figura 28 – Gráfico Consolidado em Escala Temporal

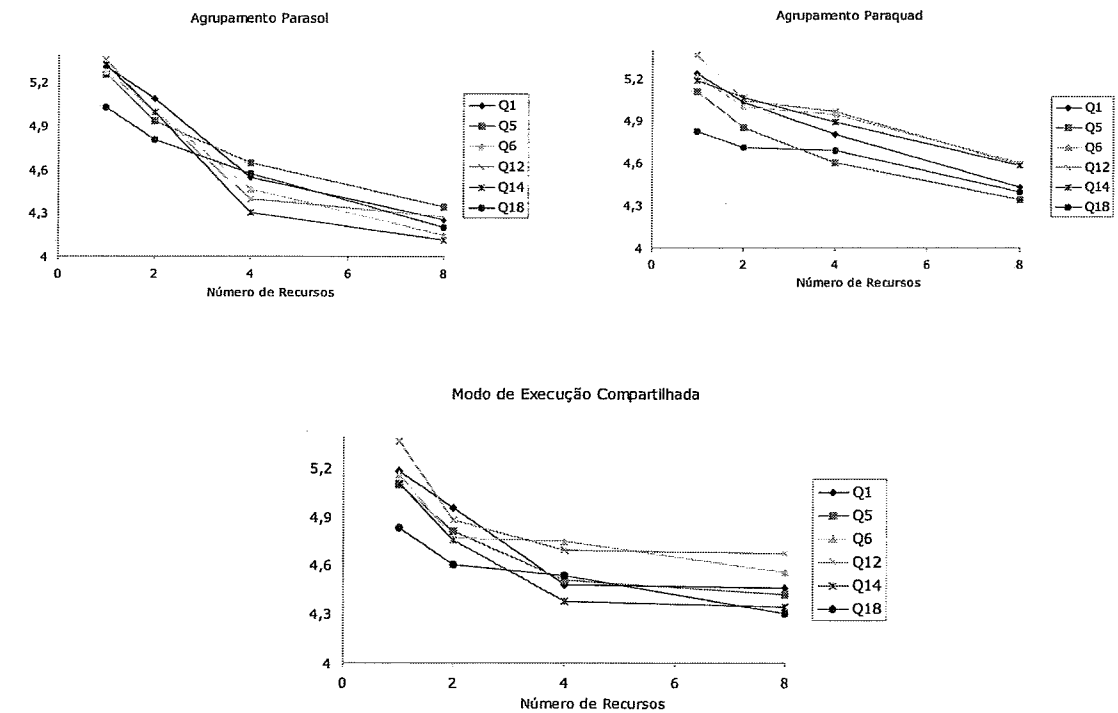

Figura 29 – Gráfico Consolidado em Escala Logarítmica

Todos os cenários apresentam desempenho similar, com uma discreta melhora durante o modo de execução compartilhada. Além disso, os resultados obtidos com o

GParGRES apresentam aceleração linear ou próxima a linear na execução das consultas, conforme mais nós são adicionados, em todas as configurações utilizadas.

Na execução de experimentos com o agrupamento paraquad, também foi possível observar que ao passarmos de dois para quatro nós utilizados, a aceleração obtida não apresentou perfil linear como o observado nos demais cenários.

Não foi possível apontar um ponto específico que justificasse este comportamento, mas podemos visualizar fatores distintos, como o comportamento do próprio ambiente de grade, dinâmico e não previsível, a alocação dos fragmentos virtuais em memória, diretamente relacionada ao fator de escala utilizado para a base de dados do *benchmark* TPC-H e a capacidade de recursos físicos e sua utilização durante os experimentos.

Todos os resultados foram obtidos sem otimizações de SGBD, rede ou máquina, o que encoraja o uso da aplicação GParGRES para a execução de consultas de apoio à decisão em grades.

Para tal, com base nos tempos estimados nas tentativas de execução com cache frio das consultas do *benchmark*, na quantidade de nós desejáveis para realização de experimentos na grade, na matriz representativa da quantidade total de experimentos, na complexidade de montagem e realização de tais experimentos e com em um fator de escala que representa uma massa de dados de maior amplitude, entendemos que seria necessário realizar um projeto de testes de aplicação com extrema cautela e que corresponde a uma análise prolongada, ou seja, que deve estender-se por um largo período de tempo.

Dessa forma, imaginamos que com um projeto específico de testes de aplicação, aliado a análise estatística adequada (desvio-padrão, médias e medianas de cada configuração e consultas associadas, por exemplo), poderíamos então apresentar um relatório formal que representasse uma análise de desempenho em larga escala.

O objetivo maior desta dissertação, que contempla a viabilização de uma aplicação mediadora acima do ParGRES, capaz de propiciar a execução de consultas de apoio à decisão, como as representadas pelo *benchmark* TPC-H, em um ambiente real de grade, foi alcançado, com a realização de experimentos iniciais e com uso de técnicas compatíveis com a infra-estrutura de serviços *web* e para grade.

O ambiente de grade envolvido na realização de experimentos com o GParGRES suportaria a utilização de maior quantidade de nós em tal processo, o que não foi realizado de acordo com as justificativas citadas acima.

Nestes experimentos, foi verificado desempenho satisfatório e com comportamento similar ao ParGRES, através de observações de aceleração linear e próxima a linear em determinados momentos.

Capítulo 5 – Conclusão e Trabalhos Futuros

Soluções para a área de banco de dados que envolvem o ambiente de grades são alvo de pesquisa atual, como no caso da criação de mediadores (ALPDEMIR, MUKHERJEE et al., 2003; ANJOMSHOAA, ANTONIOLETTI et al., 2005), no suporte ao uso de soluções relacionadas a armazéns de dados (LAWRENCE, DEHNE et al., 2007; LAWRENCE e RAU-CHAPLIN, 2006; WERHLE, MIQUEL et al., 2005) e aplicações comerciais como em (SHIMP e MIRANDA, 2005).

O suporte à execução de consultas de apoio à decisão é um dos tópicos abordados por tais soluções, e ao lidar com consultas desta natureza em tais ambientes, surgem problemas como realizar o processamento das consultas, adotar técnicas de paralelismo inter - e intra-consulta e composição de resultados de sub-consultas provenientes de cada nó da grade, de forma não-intrusiva aos SGBD, bases de dados e aplicações finais.

Com as pesquisas e possibilidades relacionadas à resolução de tais problemas, a possibilidade crescente de uso do poder computacional oferecido por ambientes de grade, distribuídos e heterogêneos, por aplicações tradicionalmente centralizadas, como as soluções para armazéns de dados, faz com que cresça também o número de usuários interessados em aplicativos que possam beneficiar-se de tais ambientes.

Consequentemente existe a tendência de observar um aumento na demanda por soluções capazes de manipular grandes massas de dados em ambientes de grade. Nesse sentido, justifica-se a pesquisa de novas iniciativas que envolvam, por exemplo, o uso do ParGRES, aplicação mediadora para prover paralelismo inter - e intra-consulta em agrupamentos de bancos de dados, como base.

Nesta dissertação abordamos os problemas associados ao processamento paralelo de consultas de apoio à decisão (OLAP) em grades computacionais, especificamente como prover paralelismo inter - e intra-consulta, de forma não intrusiva às bases de dados, SGBD associados e aplicações adjacentes, em um ambiente caracterizado por dinamismo e heterogeneidade, através do uso de técnicas atuais de criação de serviços para grade.

Para tal, propomos e concretizamos o GParGRES, concebido na forma de um conjunto de serviços para grade, compatível com o arcabouço WSRF, através da

implementação do projeto Apache Muse para o mesmo. A escolha pelo Apache Muse foi motivada por fatores como a independência de mediador de grade, a natureza de código aberto do Apache Muse e a familiaridade da linguagem utilizada no desenvolvimento do mesmo, Java, em relação ao próprio GParGRES e ao ParGRES.

Além disso, também abordamos o processamento e paralelismo das consultas em dois níveis: na grade, através de um mecanismo de fragmentação virtual simples fornecido pelo GParGRES e nos agrupamentos de PC que compõe a grade, através do mecanismo de AVP do ParGRES.

A avaliação funcional e de desempenho do GParGRES foi realizada no ambiente *Grid'5000*, plataforma configurável e escalável de grade localizada na França, através da execução de consultas do *benchmark* TPC-H, que ilustra aplicações, base de dados e consultas tipicamente OLAP, e foi executada em diferentes cenários.

Dentre as opções encontradas atualmente na literatura para suporte à execução de consultas de apoio à decisão em ambientes de grade e analisadas nesta dissertação, o GParGRES diferencia-se das mesmas por ser uma solução que abrange todos os problemas aqui listados no suporte à execução de consultas de apoio à decisão, com elevado nível de generalidade, em uma mesma camada de aplicação.

Neste sentido, reforçamos que dentre as contribuições que o GParGRES oferece, aquela que representou ao mesmo tempo o maior desafio e a maior contribuição envolveu a própria criação do conjunto de serviços, com base em padrões atuais de desenvolvimento voltado para ambientes de grade, sem que fosse necessário utilizar de mediador específico para grade, como por exemplo, o Globus Toolkit 4 (FOSTER, 2005).

O GParGRES foi desenvolvido de forma a maximizar o seu próprio potencial de portabilidade entre os diversos ambientes de grade que podemos encontrar atualmente. Não existem requisitos de aplicações ou mediador de grade específico, assim, pode-se dizer que para a execução do GParGRES, necessita-se apenas que a grade suporte a execução de programas com base em linguagem Java e a instalação do servidor Tomcat em cada um dos nós da grade que hospedará um ou mais serviços do GParGRES.

Os resultados observados nos testes iniciais de desempenho são encorajadores e tornam o GParGRES uma solução com potencial atrativo para a execução de consultas de apoio à decisão em grades. Consideramos que outro fator relevante para tal sinalização positiva com relação ao GParGRES reside no fato de conseguirmos realizar experimentos em um ambiente real de grade, sem que fosse necessário aplicar, por

exemplo, um simulador. Dessa forma, foi possível observar o funcionamento do GParGRES de forma realista.

A escolha por um arcabouço de serviços para grade e de uma implementação do mesmo caracterizada pela independência de mediador de grade mostrou-se desafiadora, pois pudemos observar que alternativas específicas para determinados mediadores de grade, como no caso do Globus Toolkit, apesar de também apresentarem dificuldades na concepção dos serviços, ao mesmo tempo, contam com maior gama de documentação, exemplos e suporte em geral.

Considera-se que o objetivo desta dissertação, prover uma solução na forma de serviços para grade, criada com base nos padrões e técnicas atuais de desenvolvimento de tais serviços, e que ofereça processamento paralelo de consultas de apoio à decisão (OLAP) em grades, foi atingido. A avaliação de funcionalidade e desempenho realizada demonstra indícios de que o GParGRES apresenta potencial para a execução de consultas de apoio à decisão, de forma similar ao ParGRES, conquanto agora no contexto de grades.

Esta dissertação está inserida no contexto da linha de pesquisa de banco de dados do PESC/COPPE/UFRJ, pois é diretamente relacionada com a pesquisa de soluções para bases de dados distribuídas, computação de alto desempenho e ambientes de grade. O GParGRES teve sua origem em outro trabalho publicado pela linha de pesquisa, o ParGRES, e também deve-se ressaltar que a cooperação entre a COPPE/UFRJ e o INRIA/Universidade de Nantes foi outro facilitador para a criação e avaliação do GParGRES em um ambiente de grade real.

Dentre os possíveis caminhos para estender o GParGRES, podemos citar, por exemplo, o suporte à replicação parcial de dados, o que pode ser interessante para bases de dados extremamente volumosas, para realizar melhor aproveitamento da capacidade de armazenamento disponível nos nós da grade. A replicação parcial de dados foi abordada por (FURTADO, LIMA et al., 2005), no caso do ParGRES. Para o GParGRES, deve-se considerar também a natureza do ambiente de grade, sua complexidade, dinamismo e heterogeneidade se comparado a agrupamentos de PC.

A adoção de técnicas de balanceamento de carga entre os nós que compõem a grade também pode ser referenciada com um possível tópico merecedor de atenção para o GParGRES. Uma abordagem que implementasse tais técnicas através do uso de metadados de desempenho dos nós poderia atuar de forma a propiciar melhor uso dos

recursos de hardware disponíveis na grade, ao minimizar o tempo ocioso de cada um dos nós envolvidos na execução de uma ou mais consultas.

Outra possibilidade interessante está no suporte a algoritmos de posicionamento, voltados para a execução de consultas de natureza *top-k* (AKBARINIA, PACITTI et al., 2007). Tais consultas envolvem a manipulação de um conjunto de listas de dados, onde cada item é ranqueado localmente em cada uma das listas em ordem decrescente e depois os mesmos itens são ranqueados em uma única lista de acordo com a pontuação geral entre todas as listas, seguindo uma fórmula matemática pré-definida para cálculo de tal pontuação. A consulta *top-k* então recupera os k elementos que apresentam o maior ranqueamento geral.

Este tópico ainda não foi explorado com extensa atenção em grades e tem como um dos principais desafios para o GParGRES a manutenção de tais listas, uma vez que enquanto em um ambiente centralizado estas são mantidas em um único nó, em um ambiente de grade, as mesmas podem estar distribuídas entre os nós do mesmo, além de podermos ter a própria base de dados pode estar fragmentada fisicamente entre os nós.

Além disso, outro tópico que possui potencial para ser explorado em conjunto com o GParGRES está relacionado a execução de fluxos de trabalho (*workflows*) científicos, que façam uso de consultas a bases de dados, o que pode ser atendido pelo GParGRES, que por sua vez beneficia-se do poder computacional distribuído do ambiente de grade.

Outra ação que também representa possibilidade interessante para o GParGRES está, por exemplo, na integração do GParGRES com as soluções OGSA-DQP ou mediadores para ambientes de grade como o Globus Toolkit, uma vez que o GParGRES é uma camada mediadora caracterizada por apresentar alto nível de generalidade.

Finalmente, observamos que existem diversas opções que corroboram o potencial do GParGRES e indiciam a possibilidade de continuidade de uso e a própria evolução do GParGRES, seja através dos tópicos aqui listados ou entre demais necessidades que remetam a soluções que possam beneficiar-se das técnicas providas pelo GParGRES.

Referências Bibliográficas

AKAL, F., BÖHM, K., SCHEK, H.-J., 2002, "OLAP Query Evaluation in a Database Cluster: a Performance Study on Intra-Query Parallelism". In: *Proceedings of the 6th. East-European Conference on Advances in Databases and Information Systems*, pp. 218-231, Bratislava, Slovakia

AKBARINIA, R., PACITTI, E., VALDURIEZ, P., 2007, "Best Position Algorithms for Top-K Queries", pp. 495-506, Vienna, Austria, September

ALPDEMIR, M. N., MUKHERJEE, A., PATON, N. W., et al, 2003, "Service-Based Distributed Querying on the Grid". In: *ICSOC 2003*, pp. 467-482, December

ANJOMSHOAA, A., ANTONIOLETTI, M., TKINSON, M. P., et al, 2005, "The Design and Implementation of Grid Database Services in OGSA-DAI. Concurrency and Computation: Practice and Experience", *Concurrency and Computation: Practice and Experience*, v. 17, pp. 357-376

ANNIS, J., ZHAO, Y., VOECKLER, M. W., et al, 2002, "Applying Chimera Virtual Data Concepts to Cluster Finding in the Sloan Sky Survey". In: *SC 2002*, pp. 1-14, Baltimore, Maryland, USA, November

BELL, W. H., BOSIO, D., HOSCHEK, W., et al, 2002, *Project Spitfire – Towards Grid Web Service Databases*

BELLATRECHE, L., KARLAPALEM, K., MOHANIA, M., 1999, "OLAP Query Processing for Partitioned Data Warehouses". In: *International Symposium on Database Applications in Non- Traditional Environments (DANTE '99)*, pp. 35-42, Kyoto, Japan

BOOTH, D., HAAS, H., MCCABE, F., et al, 2004, *Web Services Architecture*. W3C

CAPPELO, F., DESPREZ, F., DAYDE, M., et al, 2005, "Grid'5000: a large scale and highly reconfigurable Grid experimental testbed". In: *6th IEEE/ACM International Workshop on Grid Computing*, pp. 99-106, November

CARPENTER, B. E.,JANSON, P. A., 2004, "Abstract interdomain security assertions: A basis for extra-grid virtual organizations", *IBM Systems Journal*, v. 43, n. 4, pp. 689-701

CECCHET, E., MARGUERITE, J., ZWAENEPOEL, W., 2004, "C-JDBC: Flexible Database Clustering Middleware". In: *Freenix 2004: USENIX Annual Technical Conference*, pp. 9-18, Boston, USA

DAIS, 2007, "Data Access and Integration Services.". In: https://forge.gridforum.org/projects/dais-wg, Acessado em 30/10/2007.

ECLIPSE, 2008, "Eclipse". In: http://www.eclipse.org, Acessado em 02/01/2008.

ECLIPSE TPTP, 2008, "Eclipse TPTP". In: http://www.eclipse.org/tptp/home/project_info/general/whatisTPTP.php, Acessado em 02/01/2008.

EELA, 2007, "E-Infrastructure Shared Between Europe and Latin America". In: http://www.eu-eela.org, Acessado em 30/10/2007.

EGEE, 2007, "Enabling Grids for e-Science". In: http://public.eu-egee.org/, Acessado em 30/10/2007.

FOSTER, I., 2005, "Globus Toolkit 4: Software for Service-Oriented Systems", *International Conference on Network and Parallel Computing*, v. 3779, pp. 2-13

FOSTER, I., KESSELMAN, K., NICK, J. M., et al, 2002, *The Physiology of the Grid. An Open Grid Services Architecture for Distributed Systems Integration*

FOSTER, I., KESSELMAN, K., TUECKE, S., 2001, "The Anatomy of the Grid. Enabling Scalable Virtual Organizations", *International Journal of High Performance Computing Applications*, v. 15, pp. 200-222

FURTADO, C., LIMA, A. A. B., PACITTI, E., et al, 2005, "Physical and Virtual Partitioning in OLAP Database Clusters". In: *17th International Symposium on Computer Architecture and High Performance Computing*, pp. 143-150, Los Alamitos, CA,

GANGLIA, 2007, "Ganglia Distributed Monitoring System". In: http://ganglia.sourceforge.net, Acessado em 30/10/2007.

GLITE, 2007, "Lightweight Middleware for Grid Computing". In: http://glite.web.cern.ch/glite/, Acessado em 30/10/2007.

GLOBUS, 2007, "Globus Toolkit". In: http://www.globus.org/, Acessado em 30/10/2007.

GLOBUS WSRF, 2008, "GLOBUS WSRF". In: http://www.globus.org/wsrf, Acessado em 02/01/2008.

GOTTSCHALK, K., GRAHAM, S., KREGER, H., et al, 2002, "Introduction to Web Services Architecture", *IBM Systems Journal*, v. 41, n. 2

GRID'5000, 2008, "Grid'5000". In: http://www.grid5000.fr, Acessado em 02/01/2008.

HSQLDB, 2008, "HSQLDB". In: http://www.hsqldb.org/, Acessado em 02/01/2008.

KADEPLOY2, 2008, "KADEPLOY2". In: http://kadeploy.imag.fr/, Acessado em 02/01/2008.

KOTOWSKI, N., LIMA, A. A. B., PACITTI, E., et al, 2007, "OLAP Query Processing in Grids". In: *3rd VLDB Workshop on Data Management in Grids*, Vienna, Austria, September

KOTOWSKI, N., LIMA, A. A. B., PACITTI, E., et al, 2008, "Parallel query processing for OLAP in grids", *Concurrency and Computation: Practice and Experience*, n. Special Issue for 3rd VLDB Worshop on Data Management in Grids

LAWRENCE, M., DEHNE, F., RAU-CHAPLIN, A., 2007, "Implementing OLAP Query Fragment Aggregation and Recombination for the OLAP Enabled Grid". In: *4th High-Performance Grid Computing Workshop*, California, USA, March

LAWRENCE, M.,RAU-CHAPLIN, A., 2006, "The OLAP-Enabled Grid: Model and Query Processing Algorithms". In: *20th Annual International Symposium on High Performance Computing Systems and Applications*, Newfoundland, Canada, May

LIMA, A. A. B., 2004, *Paralelismo Intra-Consulta em Clusters de Banco de Dados*, Tese de Doutorado, COPPE - Universidade Federal do Rio de Janeiro.

LIMA, A. A. B., MATTOSO, M., VALDURIEZ, P., 2004, "Adaptive Virtual Partitioning for OLAP Query Processing in a Database Cluster". In: *19º Simpósio Brasileiro de Banco de Dados*, pp. 92-105, Brasília, October

MASSIE, M. N., CHUN, B. N., CULLER, D. E., 2004, "The ganglia distributed monitoring system: design, implementation, and experience", *Parallel Computing*, v. 30, pp. 817-840

MATTOSO, M., ZIMBRÃO, G., LIMA, A. A. B., et al, 2005a, "ParGRES: uma camada de processamento paralelo de consultas sobre o PostgreSQL". In: *Worskhop de Software Livre - FISL 6.0*, Porto Alegre, Brasil

MATTOSO, M., ZIMBRÃO, G., LIMA, A. A. B., et al, 2005b, "ParGRES: Middleware para Processamento Paralelo de Consultas OLAP em Clusters de Banco de Dados". In: *Sessão de Demonstrações do 20º SBBD*, pp. 19-24

MATTOSO, M., ZIMBRÃO, G., LIMA, A. A. B., et al, 2005c, *ParGRES: a middleware for executing OLAP queries in parallel.* Technical Report ES-690. PESC/COPPE

MDS4, 2008, "MDS4". In: http://dev.globus.org/wiki/MDS4 , Acessado em 02/01/2008.

MPLS, 2008, "MPLS". In: http://www.ietf.org/html.charters/mpls-charter.html, Acessado em 02/01/2008.

MUSE, 2008, "Apache Muse". In: http://ws.apache.org/muse/, Acessado em 02/01/2008.

OAR2, 2008, "OAR2". In: http://oar.imag.fr/, Acessado em 02/01/2008.

ORACLE, 2007, "Oracle Corporation". In: http://www.oracle.com, Acessado em 30/10/2007.

PACITTI, E., VALDURIEZ, P., MATTOSO, M., 2007, "Grid Data Management: open problems and new issues", *Journal of Grid Computing*, v. 5, n. 3, pp. 273-281

PATON, N. W., ATKINSON, M. P., DIALANI, V., et al, 2002, *Data Access and Integration Services on the Grid.* UK e-Science Programme Technical Report Series

POLAR, 2007, "Parallel Oriented Database". In: http://www.ncl.ac.uk/polar/, Acessado em 30/10/2007.

RENATER, 2007, "Renater". In: http://www.renater.fr, Acessado em 30/10/2007.

RÖHM, U., BÖHM, K., SCHEK, H.-J., 2002, "FAS - A Freshness-Sensitive Coordination Middleware for a Cluster of OLAP Components". In: *VLDB 2002*, pp. 754-765, Hong Kong, China, August

SACERDOTI, F. D., KATZ, M. J., MASSIE, M. N., et al, 2003, "Wide Area Cluster Monitoring with Ganglia". In: *2003 IEEE International Conference on Cluster Computing (CLUSTER 2003)*, pp. 289-, Kowloon, Hong Kong, China, December

SANTISTEBAN, M. A. N., GRAY, J., SZALAY, A. S., et al, 2005, "When Database Systems Meet the Grid". In: *2005 CIDR Conference*, pp. 154-161

SHIMP, R. G.,MIRANDA, N., 2005, *Oracle Grid Computing.* Oracle Corporation

SMITH, J., GOUNARIS, A., WATSON, P., et al, 2002, "Distributed Query Processing on the Grid". In: *Third Workshop on Grid Computing*, pp. 279-290, Baltimore, MD, USA

SMITH, J., WATSON, P., SAMPAIO, S. F. M., et al, 2000, "Polar: An architecture for a Parallel ODMG Compliant Object Database". In: *CIKM 2000*, pp. 352-359, McLean, VA, USA

SOAP, 2008, "SOAP". In: http://www.w3.org/TR/soap/, Acessado em 02/01/2008.

SOTOMAYOR, B., 2005, "The Globus Toolkit 4 Programmer's Tutorial". *University of Chicago*.http://gdp.globus.org/gt4-tutorial/

SQL92, 2008, "SQL92". In: http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt, Acessado em 02/01/2008.

TOMCAT, 2008, "Apache Tomcat". In: http://tomcat.apache.org/, Acessado em 02/01/2008.

TPCH, 2008, "TPCH-Benchmark". In: http://www.tpc.org, Acessado em 02/01/2008.

W3C, 2007, "W3C Group". In: http://www.w3.org, Acessado em 30/10/2007.

WATSON, P., 2003, *Databases and The Grid.* UK e-Science Technical Report Series.http://www.cs.ncl.ac.uk/research/pubs/books/papers/185.pdf

WEBSPHERE, 2008, "WebSphere". In: http://www.alphaworks.ibm.com/tech/wsrf4was, Acessado em 02/01/2008.

WERHLE, P., MIQUEL, M., TCHOUNIKINE, A., 2005, "A Model for Distributing and Querying a Data Warehouse on a Computing Grid". In: *ICPADS 2005*, pp. 203-209, Fukuoka, Japan, July

WSDL, 2008, "WSDL". In: http://www.w3.org/TR/wsdl20/, Acessado em 02/01/2008.

WSDL2JAVA, 2008, "WSDL2JAVA". In: http://ws.apache.org/axis/java/user-guide.html, Acessado em 02/01/2008.

WSRF, 2008, "Web Services Resource Framework". In: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf, Acessado em 02/01/2008.

XSD, 2008, "XSD". In: http://www.w3.org/XML/Schema, Acessado em 02/01/2008.